Universidad Politécnica
de Madrid

Escuela Técnica Superior de Ingenieros Informáticos

Grado en Ingeniería Informática

Trabajo Fin de Grado

Lectura Fácil: de Texto a Representación Gráfica

Autor: Rodrigo Menéndez Trejo
Tutor: María del Carmen Suárez de Figueroa Baonza

Madrid, 06 - 2025

Este Trabajo Fin de Grado se ha depositado en la ETSI Informáticos de la Universidad Politécnica de Madrid para su defensa.

Trabajo Fin de Grado
Grado en Ingeniería Informática

Título: Lectura Fácil: de Texto a Representación Gráfica

06 - 2025

Autor: Rodrigo Menéndez Trejo
Tutor: María del Carmen Suárez de Figueroa Baonza
Departamento de Inteligencia Artificial
Escuela Técnica Superior de Ingenieros Informáticos
Universidad Politécnica de Madrid

Índice general

Índice de Figuras

Índice de Tablas

Índice de Códigos

Resumen

Este proyecto se centra en hacer más fácil la comprensión de textos complejos para personas que tienen dificultades para leer o comprender lo leído. Para lograrlo, se ha diseñado una aplicación web que transforma textos en español a sus respectivas versiones adaptadas a lectura fácil, para posteriormente representarlos como un grafo visual mostrando las ideas principales y cómo se relacionan entre sí.

La lectura fácil es una forma de presentar la información de un texto de manera clara, sencilla y entendible para personas con dificultades. Existen ciertas normas concretas para adaptar textos a lectura fácil, como emplear frases cortas, evitar palabras y conceptos complejos, sustituir números grandes y porcentajes por palabras como «muchos» y «pocos», etc. Estas reglas están comprendidas en la norma *UNE 153101:2018 EX Lectura Fácil. Pautas y recomendaciones para la elaboración de documentos* [1]

Para realizar este proyecto se utilizaron técnicas de procesamiento de lenguaje natural para la simplificación del texto, se emplearon y contrastaron varios modelos de lenguaje como LLaMa 4 y herramientas especializadas en reconocimiento de entidades nombradas y extracción de relaciones para la extracción de entidades y relaciones clave. Finalmente, se diseñó una aplicación web con una interfaz sencilla que permite al usuario introducir un texto en español; este es simplificado para luego mostrar su estructura mediante un grafo.

Esta aplicación web no sólo se empleará para simplificar texto para personas con discapacidades, sino que también podrá ser usada cuando se necesite adaptar información para personas con necesidades específicas, centros educativos, extranjeros, etc. La aplicación web cuenta con herramientas adicionales como un asistente virtual capaz de explicar cualquier pregunta relacionada con el proceso de la adaptación a lectura fácil.

Abstract

This project focuses on making complex texts easier to understand for people who have difficulty reading or comprehending what they read. To achieve this, a web application has been designed that transforms Spanish texts into their respective versions adapted to easy reading, to later represent them as a visual graph showing the main ideas and how they are related to each other.

Easy reading is a way of presenting the information in a text in a clear, simple and understandable way for people with difficulties. There are certain specific rules for making texts easy to read, such as using short sentences, avoiding complex words and concepts, substituting large numbers and percentages for words such as «many» and «few», etc. These rules are included in the norm *UNE 153101:2018 EX Lectura Fácil. Pautas y recomendaciones para la elaboración de documentos* [1]

To carry out this project, natural language processing techniques were used for text simplification, several language models such as LLama 4 and specialized tools in named entity recognition and relation extraction were used and contrasted for the extraction of key entities and relations. Finally, a web application was designed with a simple interface that allows the user to enter a text in Spanish; this is simplified and its structure will be shown by means of a graph.

This web application will not only be used to simplify text for people with disabilities, but it can also be used when information needs to be adapted for people with specific needs, educational centers, foreigners, etc. In addition, this web application has special tools such as a chatbot, this virtual assistant is capable of explaining any questions related to the process of adapting text to easy reading.

Capítulo 1

Introducción

En este capítulo se detalla el contexto general de este Trabajo de Fin de Grado, incluyendo la motivación que ha llevado a cabo su desarrollo, los objetivos a alcanzar y la organización del documento.

1.1. Motivación del proyecto

Actualmente, vivimos en una sociedad donde el acceso a la información es necesario para casi todas las tareas de nuestro día a día. Sin embargo, muchas personas, ya sea por problemas de aprendizaje o aquellos que desconocen el idioma, tienen serias dificultades para comprender información escrita, visual o en audio.

La norma UNE 153101:2018 EX establece una serie de pautas y recomendaciones para la elaboración de documentos, cuyo objetivo es garantizar que todas las personas, independientemente de sus capacidades, puedan acceder y comprender la información expuesta. [1]

El objetivo principal de este Trabajo de Fin de Grado es aportar una posible solución a aquellas personas con dificultades a la hora de comprender información. Para lograrlo, se ha desarrollado una aplicación web, la cual, empleando técnicas de procesamiento de lenguaje natural (NLP), es capaz de transformar texto escrito en una representación gráfica, apoyándose en estas normas de adaptación de textos.

La finalidad de este proyecto es eliminar esta barrera de comprensión, permitiendo que más personas puedan comprender la información que les rodea.

1.2. Objetivos

El objetivo de este trabajo es crear una aplicación web capaz de convertir automáticamente textos escritos en castellano en representaciones visuales sencillas, utilizando métodos de procesamiento de lenguaje natural (NLP). Buscando así facilitar el acceso a la información a aquellas personas con dificultades para leer y/o comprender lo leído.

Para alcanzar este objetivo principal, se establecen los siguientes objetivos específicos:

- **Adaptar textos complejos a lectura fácil:** Se implementará un modelo capaz de adaptar el texto, esto lo hará basándose en las reglas establecidas por la norma UNE, con la finalidad de generar textos fáciles de leer y entender.
- **Extraer entidades y relaciones clave del texto original:** Se desarrollarán algoritmos de análisis lingüístico para identificar los elementos mas relevantes de una oración, para ello se implementará

un modelo capaz de extraer las entidades (NER) y otro capaz de extraer las relaciones (ER).

- **Diseñar un sistema capaz de convertir textos en representaciones gráficas:** A partir de la información ya estructurada, se construirá una representación visual mediante un grafo, facilitando así la comprensión del texto.
- **Desarrollar una aplicación web para la transformación automática de textos:** Se construirá una aplicación web, esta aplicación web permitirá al usuario introducir una frase en castellano que posteriormente será simplificada y mostrada como grafo de manera visual.
- **Realizar pruebas para evaluar la precisión del sistema:** Finalmente, se llevarán a cabo pruebas para verificar el correcto funcionamiento de aplicación web con todas sus integraciones anteriores.

1.3. Estructura del Documento

La estructura del documento, a partir de este capítulo, se divide en cinco bloques principales, cada uno centrado en una fase clave del desarrollo de este proyecto. La estructura es la siguiente:

- **Capítulo 2. Trabajo relacionado y estado de la cuestión:** En este capítulo se presentan los principales conceptos relacionados con la lectura fácil, el procesamiento del lenguaje natural y la representación gráfica de textos. También se exponen trabajos relacionados que han servido como referencia.
- **Capítulo 3. Desarrollo:** Se describe el proceso del diseño y la implementación de la solución propuesta a la problemática explicada en el capítulo1.
- **Capítulo 4. Resultados:** Se analizan los principales resultados obtenidos durante el desarrollo del proyecto.
- **Capítulo 5. Conclusiones y lineas futuras:** En este último capitulo se presentan las conclusiones del trabajo, las limitaciones encontradas, el impacto de los resultados obtenidos y posibles lineas futuras de mejora.

Capítulo 2

Trabajo relacionado y estado de la cuestión

En este capítulo se comentan los temas más importantes relacionados con el proyecto y que han servido como base para su desarrollo. Primero se explica qué es la lectura fácil, cómo surgió y por qué es tan importante. También se revisan las normas que regulan la adaptación de textos a lectura fácil, especialmente la norma UNE en España. Posteriormente, se presentan las herramientas y modelos actuales que permiten adaptar textos automáticamente y extraer información clave de ellos, como nombres, lugares o relaciones; además, de cómo representar esa información en forma de grafo y herramientas similares a este proyecto. Por último, se comentan las técnicas más importantes para entrenar a los modelos de lenguaje y hacer que funcionen de manera más precisa.

2.1. Lectura fácil

2.1.1. Definición y orígenes

La lectura fácil es una metodología de adaptación de textos cuyo propósito es hacer que la información sea más comprensible para aquellas personas con dificultades de lectura o comprensión. Consiste, principalmente, en adaptar la información, usando un vocabulario sencillo, explicando los conceptos técnicos, oraciones cortas y apoyos visuales, como imágenes y gráficos, para facilitar la comprensión del texto. En resumen, se trata de expresar las ideas de una manera clara y precisa, evitando palabras complejas, metáforas, números ordinales, etc. [1].

La técnica de lectura fácil surgió en Suecia en los años 60, cuando se identificó la necesidad de hacer que la información fuera más asequible para personas nuevas a la lengua y personas con discapacidades cognitivas. El primer libro en formato lectura fácil se publicó en 1968 en Suecia, [2] y posteriormente acabó extendiéndose por toda Europa una vez vistos sus beneficios, ya que beneficiaba a personas mayores, con bajo nivel educativo, inmigrantes aprendiendo el idioma y a personas con discapacidades cognitivas [2].

2.1.1.1. Contexto social y nivel lector en la población española

En el año 2013, la OCDE (Organización para la Cooperación y el Desarrollo Económicos) publicó el informe *PISA para adultos*; en este se evaluaron las competencias lectoras de personas entre 16 y 65 años en 23 países. España obtuvo una puntuación media de 252 puntos, situándose en el penúltimo lugar, solo por delante de Italia [1].

Este informe identifica seis niveles de comprensión lectora, siendo el nivel «1» el más bajo y el nivel «5» el más alto. En España, la media se sitúa en el nivel 2; esto significa que estas personas pueden realizar

tareas básicas de lectura, pero encuentran dificultades con textos complejos o muy densos [1].

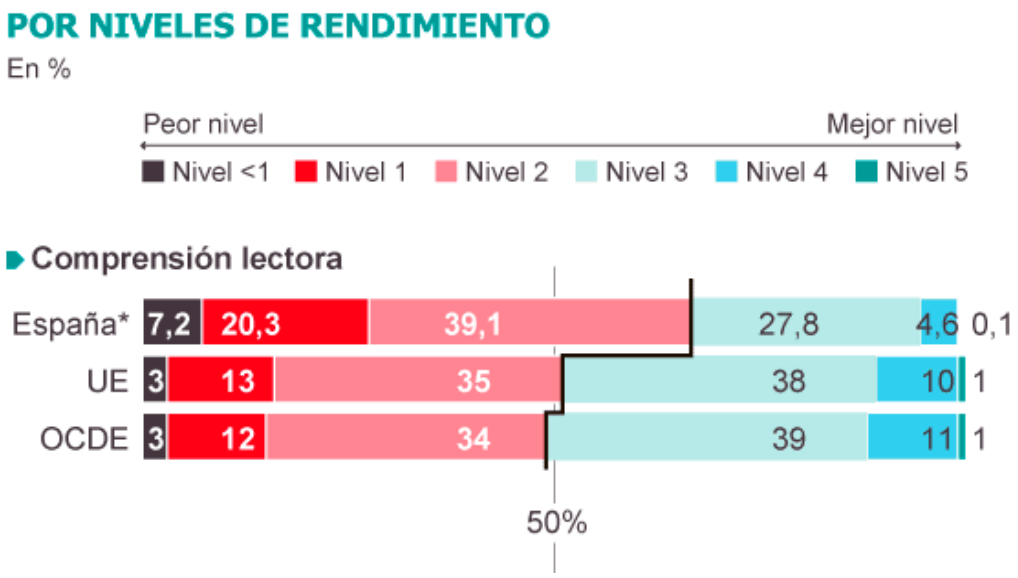


Figura 2.1: Distribución de la comprensión lectora en adultos en España, la UE y la OCDE. Fuente: *PISA para adultos, 2013*. Imagen adaptada de El País.

Como se puede observar en la Figura 2.1, más del 27 % de los adultos en España se encuentran en los niveles más bajos de comprensión (nivel 1 e inferior), y solo un 4.7 % alcanza los niveles más altos (niveles 4 y 5) [1]. Esto significa que muchas personas tienen dificultades para:

- Extraer ideas importantes de un texto.
- Entender textos y artículos densos.

Además, según datos del INE (Instituto Nacional de Estadística), en España hay aproximadamente 700.000 personas que son consideradas analfabetas funcionales [1]. Todos estos datos demuestran la necesidad de un sistema capaz de adaptar de manera automática frases y oraciones, facilitando el acceso a la información a estas personas.

2.1.2. Estándares y normativas

Con la llegada de la lectura fácil, las organizaciones europeas establecieron una serie de directrices para que los criterios de lectura fácil fuesen los mismos. En 1988 *Inclusión Europa* publicó las primeras Directrices Europeas de Lectura Fácil, conocidas como *Información para todos*, que recogen las normas para redactar documentos fáciles de leer y entender [3]. Estas directrices dan una orientación general sobre cómo redactar documentos accesibles.

En España, esas directrices fueron adaptadas y ampliadas en la norma UNE 153101:2018 EX, titulada *Lectura Fácil. Pautas y recomendaciones para la elaboración de documentos* [4]. Esta norma fue desarrollada por entidades como *Plena Inclusión* y la *Asociación Lectura Fácil*, y representa el estándar oficial para la adaptación de textos en español.

La norma UNE establece tres principios clave:

- Uso de lenguaje claro, frases cortas, evitar tecnicismos y estructuras complejas.
- Emplear un formato limpio al igual que visual y preciso.

- Revisión del texto por personas con dificultades lectoras, asegurando su comprensión.

A diferencia del resto de normativas europeas, comentadas anteriormente, la norma UNE está adaptada al español, por lo que se considera la principal fuente de referencia para este Trabajo de Fin de Grado.

2.1.3. Principales retos

La lectura fácil busca adaptar textos para que sean comprensibles por personas con dificultades para leer o comprender; sin embargo, realizar este proceso de manera automática puede resultar complejo debido a los siguientes desafíos:

- **Adaptación del texto sin pérdida de significado:** Uno de los principales desafíos es transformar el texto original, siguiendo las pautas de lectura fácil, sin modificar su significado inicial. Al aplicar múltiples transformaciones sobre una frase (léxicas, sintácticas o semánticas) existe la posibilidad de eliminar información importante o el significado del mensaje original.
- **Falta de datos paralelos en español:** Actualmente existe una carencia de corpus que contengan versiones originales y adaptadas de los mismos textos, especialmente en lengua española. Esta limitación dificulta el entrenamiento y evaluación de modelos automáticos específicos para tareas de adaptación. Como señala Štajner (2021), *"the lack of large high-quality TS datasets for training ATS systems for particular target populations"* ha impedido el desarrollo de sistemas personalizados y eficaces. Los pocos recursos disponibles *"do not provide enough material for training ATS models"* [5].
- **Limitaciones de los sistemas automáticos actuales:** Muchos sistemas existentes de adaptación realizan las transformaciones sin tener en cuenta el contexto del texto. En español, existen herramientas, como el proyecto *Simplext*, que aplican transformaciones predefinidas (por ejemplo, dividir oraciones complejas, sustituir palabras difíciles), pero estos sistemas son poco flexibles y no cubren todos los casos [6]. Por otro lado, los modelos modernos de lenguaje permiten una mayor adaptabilidad y comprensión contextual, aunque también pueden generar errores conocidos como *alucinaciones*, añadiendo información irrelevante en el texto original [7].

2.2. Procesamiento de Lenguaje Natural

Para lograr la adaptación de un texto complejo a uno más comprensible, se emplean diferentes técnicas de Procesamiento de Lenguaje Natural (PNL). En este apartado revisamos algunas técnicas clave para lograrlo: adaptación automática de texto, reconocimiento de entidades nombradas (NER, por sus siglas en inglés: *Named Entity Recognition*) y extracción de relaciones, entre otras. Estas técnicas nos permiten analizar un texto y extraer los elementos necesarios para la representación gráfica.

2.2.1. Adaptación automática de texto

Llamamos adaptación de texto a la tarea de modificar el contenido de una oración para conseguir una más sencilla de leer y comprender, manteniendo el significado original de la oración. En el Procesamiento de Lenguaje Natural, esta adaptación se aplica en varios niveles: en primer lugar, se realiza una simplificación léxica; en esta etapa se reemplazan palabras técnicas por sinónimos u otras palabras más comunes; y en segundo lugar, una simplificación sintáctica, reestructurando las oraciones complejas en oraciones más cortas. *Bott y Saggion (2012)* desarrollaron un sistema de simplificación automática de un texto en español, orientado a personas con discapacidades cognitivas. Este sistema transformaba las oraciones complejas en varias oraciones simples, reduciendo así la dificultad de la lectura [8].

Este proyecto, llamado *Simplext*, siguió el siguiente enfoque: primero identificaba en el texto las partes complejas (oraciones subordinadas, metáforas, etc.); posteriormente, aplicaba reglas para simplificarlas, sustituyendo términos difíciles por unos más sencillos [8]. Sin embargo, este campo se enfrenta a desafíos como: evitar cambiar o perder el significado a la hora de realizar la simplificación, manejar correctamente expresiones ambiguas, metáforas, etc.

2.2.2. Modelos de adaptación automática de texto

Los modelos de lenguaje de gran tamaño, conocidos como *LLMs* (por sus siglas en inglés: *Large Language Models*), son un tipo de programa de inteligencia artificial entrenados con grandes cantidades de texto para generar, resumir, traducir o adaptar lenguaje natural. Estos modelos aprenden patrones y relaciones entre las palabras, entendiendo el contexto para producir una salida adecuada [9]. Esto los convierte en una herramienta clave para las tareas de procesamiento de lenguaje natural.

Para la simplificación automática de textos se han evaluado diferentes modelos de lenguaje de última generación. A continuación, se resume cada uno:

- **Modelos de Cohere:** Cohere es una organización que ofrece diversos modelos de lenguaje a través de un servicio API. Los modelos Cohere de generación de texto están entrenados en diferentes lenguajes [10]. Cohere tiene modelos como *Embed* que soportan más de 100 idiomas y proyectan texto de diferentes lenguas en un mismo espacio semántico [10].

- **Salamandra:** Es un modelo open-source desarrollado por el *Barcelona Supercomputing Center* (BSC). Salamandra fue entrenado desde cero con datos de 25 idiomas europeos. Hay tres tamaños de modelos: 2.7 y 40 billones de parámetros [11]. El objetivo de Salamandra es proporcionar modelos centrados en las lenguas europeas reduciendo así la dependencia de los modelos en inglés [11]. Salamandra, a diferencia de otros modelos, es completamente gratuito y puede ser descargado de manera local, demostrando así que es viable tener LLMs locales en español y con buenos resultados.

- **Modelos de OpenAI:** Estos modelos han marcado el estado del arte en comprensión y generación de texto. Por ejemplo, OpenAI reporta que GPT-4 obtuvo una puntuación en el top 10 al hacer un examen de licenciatura de derecho de EE.UU. [12], demostrando así su capacidad para razonar. No obstante, OpenAI también tiene sus inconvenientes, al ser un servicio de pago implica un coste por uso y no se tiene el control completo del modelo ni tampoco el de los datos enviados a la API, además GPT-4, a pesar de ser extremadamente capaz en la mayoría de tareas, su acceso está limitado.

- **Modelos de Groq:** Groq es una plataforma y servicio en la nube que ofrece modelos de LLMs con muy poco latencia, se reporta que los modelos de Groq han sido 18 veces más veloces que otros modelos [13]. Groq ofrece acceso completamente gratuito a modelos populares como LLaMA-4, GPT-J, Mistral, etc. Dentro de los modelos ofrecidos por Groq cabe a destacar LLaMA (Large Language Model desarrollado por Meta AI) es de la familia de modelos de lenguaje de código abierto de Meta, este modelo se ha considerado para este proyecto, en particular su versión LLaMA 4, esta versión, lanzada en abril de 2025, representa el estado del arte en modelos de acceso gratuito [14]. LLaMA permite, bajo una licencia específica, modificar los pesos de sus modelos, consiguiendo que los usuarios puedan realizar mejoras sobre la base de LLaMA. LLaMA 4 es uno de los modelos open-source más eficientes, con el mejor rendimiento tanto en razonamiento como en otras tareas [14].

2.2.3. Reconocimiento de entidades

El reconocimiento de entidades nombradas es una tarea que consiste en identificar las entidades en un texto y clasificarlas en categorías (personas, organizaciones, lugares, fechas, cantidades, etc.) [15]. Por ejemplo, la frase «Rodrigo compró 300 acciones de Apple en 2006», un sistema de reconocimiento de entidades nombradas identificaría «Rodrigo» como entidad Persona, «Apple» como entidad Organización y «2006» como entidad Fecha [15].

Actualmente, gracias a los avances en aprendizaje profundo (redes neuronales, BERT, etc.), los sistemas de reconocimiento de entidades nombradas han logrado una alta precisión [15]. Para el idioma castellano, el reconocimiento de entidades se ha conseguido empleando grandes corpus anotados para el entrenamiento de modelos. Algunas herramientas como spaCy o Stanford NLP incluyen modelos pre-entrenados para la extracción de entidades de un texto. Además de estas herramientas, los modelos de lenguaje de gran tamaño como GPT, LLaMA o T5 demuestran un gran rendimiento en estas tareas, sin necesidad de un entrenamiento adicional, ya que a partir del contexto son capaces de reconocer más entidades que los modelos clásicos. Mediante técnicas de prompting, explicadas en la sección 2.6, estos modelos son capaces de identificar entidades de manera precisa incluso en contextos nuevos, lo que los convierte en una gran alternativa o una ayuda adicional para los modelos clásicos [16].

2.2.4. Extracción de relaciones

La extracción de relaciones (Relation Extraction, RE) es la tarea de identificar las relaciones semánticas que existen entre las entidades de un texto [17]. Este proceso, el cual se realiza una vez que se han detectado todas las entidades, busca las relaciones que unen las entidades y sus tipos. Actualmente se emplean modelos semi-supervisados o supervisados, entrenados en corpuses anotados con relaciones, o incluso modelos de lenguaje, los cuales generan tripletas directamente a partir de un texto [17].

Para la generación de grafos, la extracción de entidades es uno de los aspectos fundamentales, ya que es necesario obtener todas las relaciones entre entidades para formar las aristas del grafo.

2.2.5. Herramientas PLN: spaCy, Stanza y TextRazor

A la hora de reconocer entidades y relaciones, cabe destacar herramientas de procesamiento de lenguaje natural como spaCy, Stanza y TextRazor; estas herramientas destacan en tareas de extracción de entidades y relaciones y son reconocidas por su capacidad para el análisis en español.

- **spaCy:** Es una biblioteca de código abierto muy utilizada para el procesamiento de lenguaje natural. Proporciona modelos pre-entrenados capaces de realizar una tokenización (dividir el texto en unidades más pequeñas como palabras, signos de puntuación o números, que luego pueden ser empleadas por el modelo), análisis sintáctico y reconocimiento de entidades nombradas, entre otras funciones [18]. Destaca por su velocidad y exactitud, además incluye modelos específicos por idioma; en el caso de los modelos en español, estos están entrenados con el corpus *AnCora*, diseñado para tareas como identificación de sustantivos, verbos, entidades, etc.

- **Stanza:** Anteriormente conocido como Stanford NLP, es una librería desarrollada por la universidad de Stanford que implementa modelos neuronales para el análisis lingüístico en más de 70 idiomas [19]. Emplea redes neuronales y Transformers, entrenados con los corpus universales garantizando así una mayor eficiencia [19].

  Stanza, al igual que spaCy, es capaz de reconocer las entidades y clasificarlas, dividir el texto en oraciones y tokens. *Qi et al. (2020)* reportan que Stanza alcanza rendimientos superiores, en algunos casos, a otras herramientas de análisis sintáctico en español [19].

- **TextRazor:** Es un servicio API para el análisis semántico del lenguaje natural, a diferencia de spaCy y Stanza, TextRazor incorpora una base de conocimiento para entidades, extracción de temas y la clasificación de estos [20]. En resumen, TextRazor es capaz de relacionar cada palabra con un identificador único en base al contexto(por ejemplo, relacionar «León» a una provincia de España, y siendo capaz de distinguirla de «León» como animal) [20].

2.3. Generación de grafos a partir de texto

Uno de los aspectos más importantes para la adaptación de un texto a lectura fácil es la generación de un grafo visual a partir del texto, con el objetivo de representar la información de una manera más organizada y estructurada, es decir, visualmente más simple. Esto es esencial para personas que tienen dificultad para leer y comprender textos.

2.3.1. Grafos de conocimiento

La construcción de un grafo de conocimiento se basa en las técnicas mencionadas en la sección 2.2.3 y 2.2.4, donde, a partir de una serie de entidades y relaciones, se construyen los nodos (entidades) y las aristas (relaciones). Una vez obtenidas estas tripletas (entidad-relación-entidad) podemos construir grafos visuales e interactivos.

Entre las herramientas más utilizadas para la construcción de grafos encontramos bibliotecas como *Pyvis*, que permite generar grafos interactivos y emplear funciones de fuerza para que los nodos no se superpongan. También existen aplicaciones web como *MyMap AI* o *ConceptMap AI*, que permiten representar y explorar nodos y relaciones de manera visual y dinámica [21].

Esta representación gráfica facilita la comprensión del contenido del texto, especialmente en aquellos contextos donde es necesario extraer información de manera rápida.

2.4. Herramientas para el desarrollo de aplicaciones web interactivas

El desarrollo de una aplicación web interactiva es la tarea más importante de este proyecto, gracias al diseño de una página web el usuario es capaz de ver el grafo resultante al introducir un texto. Entre las herramientas más utilizadas para el desarrollo de páginas web destaca *Streamlit*, una biblioteca open-source basada en Python que permite crear aplicaciones web de forma rápida y sencilla. Streamlit está orientado para crear interfaces sin necesidad de una gran experiencia previa. Permite desplegar botones, gráficas, entradas de texto y diferentes tipos de visualizaciones [22].

Además de Streamlit, existen otras herramientas como:

- **Dash (Plotly)**: Tiene un framework más completo, basado en Flask, con capacidad para desarrollar aplicaciones interactivas además es capaz de una mayor personalización de la página web.
- **Voila**: Convierte notebooks de Jupyter en dashboards interactivos sin mostrar el código.

Estas herramientas permiten integrar modelos de lenguaje, procesamiento de texto y visualización de datos en páginas web desde cualquier navegador. De esta forma, el usuario final puede visualizar todas las herramientas de la página web visitando un enlace.

2.5. Sistemas y plataformas similares: GraphGPT

Un claro ejemplo de una herramienta capaz de convertir textos complejos en grafos es GraphGPT. Es un proyecto open-source, el cual utiliza modelos de lenguaje de OpenAI para interpretar un texto en castellano y producir un grafo de conocimiento [23]. Esta herramienta toma un texto en castellano y genera una visualización gráfica de las entidades y las relaciones de ese texto. Esto lo logra empleando modelos como GPT-3, junto a un prompt diseñado para formatear la salida como un grafo. Según la documentación oficial, GraphGPT fue diseñado como un «toy project» experimental para demostrar cómo un modelo GPT-3 puede crear un grafo desde un texto dado [23].

Entrando más en detalle, GraphGPT funciona a través de few-shot prompting: al modelo se le proporcionan algunos ejemplos de texto y una representación final en JSON de un grafo (es decir, una lista de nodos y relaciones). Al recibir un texto, se hace una llamada a la API de OpenAI y GPT-3 produce directamente el JSON con los nodos y aristas etiquetados. Este JSON es luego empleado para dibujar grafos interactivos (en este caso, usando la biblioteca react-graph-vis) [23].

2.5.1. Limitaciones de GraphGPT

Al ser un proyecto construido sobre un modelo de lenguaje, GraphGPT tiene algunas limitaciones. La primera de ellas es la latencia: cada llamada a la API de OpenAI tarda varios segundos, donde el tiempo de respuesta ronda los 20 segundos en los mejores casos[23]; otra limitación importante es la API de OpenAI, esta necesita una clave API válida y tiene un coste por cada llamada realizada [24].

2.5.2. Comparativa con el Trabajo de Fin de Grado

El trabajo de Fin de Grado comparte el mismo objetivo que GraphGPT, transformar un texto complejo en una representación gráfica comprensible. Sin embargo, hay varias diferencias:

- **Lectura fácil vs. extracción genérica de GraphGPT:** A diferencia de GraphGPT, que su salida no está adaptada a lectura fácil, este Trabajo de Fin de Grado simplifica un texto a partir de una serie de directrices recogidas en la *norma UNE 153101:2018 EX Lectura Fácil. Pautas y recomendaciones para la elaboración de documentos*[1].
- **Tecnología subyacente:** GraphGPT se apoya en un *Large Language Model* (GPT-3) y en prompt engineering, devolviendo, a partir de una llamada, las entidades y relaciones del texto. Este proyecto combina LLMs para la adaptación de texto y para la extracción de entidades y relaciones, aplicando varios prompts consecutivos para no depender únicamente de una sola llamada.
- **Limitaciones de contenido:** Una de las limitaciones más importantes de GraphGPT es el uso de la API de OpenAI, para hacer llamadas a modelos como GPT-3 es necesario hacer un deposito inicial, luego se cobra en función del número de tokens llamados con la API [24].

En conclusión, GraphGPT demuestra ser capaz de construir grafos visuales a partir de un texto dado empleando LLMs. Sin embargo, sus limitaciones (tiempo de respuesta, depende de una API de pago, etc.) son áreas que este proyecto va a mejorar. En cuanto al alcance, GraphGPT es una herramienta de uso general, que no adapta el texto introducido en base a las normas de lectura fácil recogidas por la norma UNE [1].

2.6. Técnicas de Prompting para LLMs

Como ya se comentó en secciones anteriores, el prompting es una forma de mandar instrucciones a modelos de lenguaje. Estas técnicas de prompting son esenciales para que los modelos LLM funcionen como esperamos y para que puedan desarrollar un sistema de adaptación de un texto a lectura fácil. A continuación, se detallan las técnicas más relevantes:

- **Zero-Shot:** Consiste en proporcionar la instrucción al modelo sin emplear ejemplos adicionales, esta técnica se suele emplear para poner a prueba el conocimiento pre-entrenado del modelo y la capacidad de este para seguir instrucciones literales [25]. Sin embargo, esta técnica solo resulta útil para aquellas tareas sin una necesidad de respuesta compleja, es decir, sirve como línea base.
- **Few-Shot:** Aquí se le proporciona al modelo uno o varios ejemplos de la entrada y salida esperada, esta técnica resulta ser muy efectiva ya que el modelo ve como debería salir la salida y ajusta mucho mejor su estilo de salida, limitándose a lo aprendido en los ejemplos [25]. Estudios previos (Brown et al., 2020) ya mostraban que los LLM aprenden patrones a partir de pocos ejemplos en el contexto [25].
- **Chain-of-Thought, CoT:** Esta técnica le pide al modelo razonar paso a paso, dividiendo el problema en diferentes secciones, esta técnica es especialmente buena para ver por qué el modelo cambia algo que no necesita cambiar. Una de las principales desventajas del CoT es que necesita un prompt más largo y la respuesta también lo es. Esta técnica no funciona bien en los modelos de lenguaje pequeños, ya que devuelven un razonamiento poco fiable [25].
- **Role prompting:** Consiste en indicar al LLM que asuma un cierto rol o estilo al responder [25]. Esta técnica resulta muy eficaz en diversos temas, ya que de esta forma el modelo no añade información adicional innecesaria y tampoco se desvía del tema. Esta modalidad se puede combinar con otras técnicas de prompting, consiguiendo una respuesta mucho más adecuada y certera.
- **Otros refinamientos:** Cabe a destacar otras formas de hacer prompts, como por ejemplo exigir una estructura concreta como un JSON, asegurando así una salida parseable, se pueden usar restricciones y verificaciones en el prompt: por ejemplo, «en el caso de encontrar un número grande (200, 300, etc.) sustitúyelo por muchos o pocos». Estas indicaciones buscan forzar límites [25].

A pesar de todas estas técnicas de prompt, los modelos pueden desviarse y no ser completamente efectivos, por ello, se puede diseñar un pipeline mediante el cual, tras hacer una primera iteración con un modelo de lenguaje, la respuesta ya procesada de un previo modelo se le pasa como input a otro modelo, de esta manera nos aseguramos de que la salida final cumpla con el prompt especificado; esto es una técnica muy recomendada en prompt engineering.

Capítulo 3

Desarrollo

En este capítulo se detalla cómo se ha construido el proyecto propuesto, desde las primeras decisiones de modelos a usar hasta su implementación final. Se explican las fases principales del desarrollo: la adaptación del texto a lectura fácil, la extracción de entidades y relaciones, la creación del grafo visual y, finalmente, el diseño de la aplicación web. También se comentan las herramientas empleadas, los modelos y cómo se ha organizado el trabajo.

3.1. Simplificación de textos a lectura fácil

3.1.1. Normativas y principios de lectura fácil aplicados

La Lectura Fácil es una metodología regulada en España mediante la norma *UNE 153101:2018 EX*, que establece los criterios para crear documentos accesibles a personas con dificultades de comprensión lectora. Esta norma incluye una serie de pautas a seguir para adaptar un texto a lectura fácil.

En este Trabajo de Fin de Grado, se ha tomado como referencia la norma UNE, en especial aquellas reglas que pueden aplicarse automáticamente usando modelos de lenguaje, es decir, aquellas reglas que pueden ser instruidas a un modelo de lenguaje mediante un prompt. A continuación, se mencionan las reglas de simplificación que se han aplicado a lo largo del desarrollo.

- **Empleo de palabras sencillas:** Se evita el uso de tecnicismos, palabras que acaban en «mente» y palabras que exageren y terminen en «isimo».
- **Explicación de términos complejos:** En caso de que se necesite usar una palabra compleja (por ejemplo, términos médicos), esta debe de ser explicada.
- **División en frases cortas:** Se debe emplear frases cortas y breves, cada una de estas frases debe contener una única idea.
- **Formato accesible para los números:** Se evitan los números ordinales, fracciones, porcentajes o cifras grandes, sustituyéndolos por expresiones más comprensibles. Los números deben escribirse con cifras excepto aquellos valores grandes que pueden escribirse con letra o con una comparación.
- **Evitar abreviaturas y siglas:** Se eliminan o sustituyen las siglas y abreviaturas por su forma completa.
- **Evitar el uso de voz pasiva:** Se aconseja el uso de verbos simples y evitar la voz pasiva siempre que se pueda. Se puede emplear el presente para facilitar la comprensión del texto.

- **Evitar el uso de frases negativas:** Siempre que se pueda, se debe utilizar frases positivas en vez de frases negativas.
- **Evitar el uso de dos o más verbos seguidos:** Las oraciones con estructuras verbales compuestas se simplifican a una sola acción por frase.

3.1.2. Implementación basada en LLM. Comparativa de modelos

En este proyecto se usa una implementación basada en modelos de lenguaje (LLM, Large Language Model) para la simplificación del texto. A lo largo del desarrollo se han evaluado diferentes plataformas que ofrecen acceso a diferentes modelos de lenguaje, de manera gratuita. En concreto, se usaron y compararon tres servicios centrados en la generación de texto en español: Cohere, Salamandra y Groq. Estos modelos han sido comparados empleando la técnica zero-shot prompt, es decir, a los modelos se les proporcionó una instrucción, simplificar empleando las normas de la UNE, sin emplear ejemplos adicionales. De esta manera se pone a prueba los conocimientos pre-entrenados del modelo.

3.1.2.1. Cohere

El primer modelo que se puso en práctica fue el modelo *command-r-plus* de Cohere. Para su funcionamiento, se hace una llamada empleando la API de Cohere con un prompt específico, instruyendo al modelo a actuar como un experto en lectura fácil y simplificando el texto con algunas de las normas explicadas en la sección 3.1.1. En el Listing 3.1 se muestra un ejemplo de llamada:

```
co = cohere.Client("API_KEY")
texto = "La paella valencia tiene 300gr. de arroz..."
response = co.generate(
    model="command-r-plus",
    prompt=(normas + texto),
    max_tokens=150,
    temperature=0.4,
)
print(response.generations[0].text)
```

Listing 3.1: Llamada a la API de Cohere

La Tabla 3.1 muestra una entrada de prueba y la respuesta generada por el modelo.

Texto original	**Texto simplificado por Cohere**
La paella valencia tiene 300 gr. de pollo, 350 gr. de arroz y ha sido preparada por el mejor cocinero de Madrid.	La paella valenciana tiene pollo y arroz. El pollo pesa 300 gramos. El arroz, 50 gramos más: 350 gr. La ha hecho el mejor cocinero de Madrid.

Tabla 3.1: Ejemplo de simplificación usando el modelo *command-r-plus* de Cohere.

Como se puede observar en esta tabla, el modelo es capaz de realizar de manera correcta muchas simplificaciones; no obstante, introduce información adicional que puede llegar a ser confusa.

El modelo *command-r-plus* de Cohere, combinado con un buen prompt, es capaz de devolver casi siempre los resultados esperados. Sin embargo, se observaron algunos problemas:

- No siempre sustituía tecnicismos o abreviaciones como «gr.».
- Al ser un servicio de pago, la versión gratuita tiene varias limitaciones; el número de restricciones para usar la API es muy elevado, permitiendo hacer, únicamente, cinco llamadas por minuto, restringiendo severamente su uso.
- Al tener esta limitación en la API, el modelo no puede aplicar las reglas de simplificación en varias llamadas y tendría que hacerlo en una única, lo que aumentaría la posibilidad de error.
- No aplicaba todas las normas de lectura fácil y, en algunos casos, generaba alucinaciones, es decir, generaba información incorrecta o inventada.

Dado el comportamiento poco predecible del modelo y las pocas llamadas que se podían hacer, se descartó Cohere como opción principal y se pasó al siguiente modelo, Salamandra.

3.1.2.2. Salamandra

Se probó también un modelo open-source especializado en el español, *salamandra-7b*. Este modelo se ejecutó localmente usando la librería transformers de Hugging Face. Para este modelo se implementó el mismo prompt utilizado con Cohere, observando así las diferencias entre ambos. En el Listing 3.2 se muestra un ejemplo de llamada:

```
tokenizer = AutoTokenizer.from_pretrained("BSC-LT/salamandra-7b")
model = AutoModelForCausalLM.from_pretrained("BSC-LT/salamandra-7b")

inputs = tokenizer(prompt + texto, return_tensors="pt")
outputs = model.generate(**inputs, max_new_tokens=150)
print(tokenizer.decode(outputs[0], skip_special_tokens=True))
```

Listing 3.2: Llamada al modelo de Salamandra

La Tabla 3.2 muestra un ejemplo del resultado generado por *salamandra-7b* aplicando el mismo prompt que con Cohere.

Texto original	**Texto generado por Salamandra**
La paella valencia tiene 300 gr. de pollo, 350 gr. de arroz y ha sido preparada por el mejor cocinero de Madrid.	El plato valenciano contiene 1 kg de ingredientes y está elaborado por un chef madrileño.

Tabla 3.2: Ejemplo de salida generada por el modelo *salamandra-7b*.

Como se puede observar, los resultados, aplicando exactamente los mismos parámetros e instrucciones, son muy diferentes. A pesar de que salamandra-7b es un modelo especializado en español, los resultados obtenidos fueron muy poco favorables; entre sus principales problemas destacan:

- El tiempo de respuesta es mucho más lento que cualquier llamada a un modelo mediante API.
- Las salidas de este modelo, a pesar de implementar varias técnicas de prompting mencionadas en la sección 2.6, eran incoherentes y el modelo sufría muchas alucinaciones, tal y como se puede observar en la tabla 3.2.
- El modelo no es capaz de aplicar las simplificaciones necesarias para conseguir un texto adaptado a lectura fácil.

En consecuencia, se descartó salamandra y se pasó al siguiente, Groq.

3.1.2.3. Groq

Tras probar los modelos anteriores y ver los problemas que tenían, se decidió emplear los modelos ofrecidos por la plataforma Groq, en concreto el modelo Text-to-Text denominado *meta-llama/llama-4-maverick-17b-128e-instruct*, por su velocidad de respuesta y por ser un modelo de última generación. En primer lugar, se evaluó, al igual que en el resto de casos, la simplificación original obtenida por el modelo. En el Listing 3.3 se muestra un ejemplo de llamada:

```
    client = Groq()
    completion = client.chat.completions.create(
    model="meta-llama/llama-4-maverick-17b-128e-instruct",
    messages=[prompt + texto],
    temperature=1,
    max_completion_tokens=1024,
    top_p=1,
    stream=True,
    stop=None,
)
for chunk in completion:
    print(chunk.choices[0].delta.content or "", end="")
```

Listing 3.3: Llamada a la API de Groq

La Tabla 3.3 muestra un ejemplo del resultado generado por *LLaMA-4* aplicando el mismo prompt.

Texto original	**Texto generado por LLaMA-4**
La paella valencia tiene 300 gr. de pollo, 350 gr. de arroz y ha sido preparada por el mejor cocinero de Madrid.	La paella valenciana tiene 300 gramos de pollo. La paella valenciana tiene 350 gramos de arroz. La hace el mejor cocinero de Madrid.

Tabla 3.3: Ejemplo de salida generada por el modelo *LLaMA-4*.

Como se puede observar, cumple con los principios explicados en la sección 3.1.1 y los resultados obtenidos superaron en creces a los producidos por los modelos de Cohere y Salamandra.

Posteriormente, se probó una segunda versión basada en salidas estructuradas en formato JSON. Este enfoque añadió ventajas importantes, como una salida estructurada para una mayor facilidad para luego analizarlo.

Ambas versiones, tanto la salida directa como la salida estructurada, demostraron ser muy efectivas. Sin embargo, la versión JSON demostró ser mucho más robusta, ya que permitía más control. Además, el modelo se ceñía únicamente al JSON especificado y no producía alucinaciones fuera de lugar.

3.1.2.4. Comparativa final de los modelos usados

A continuación, se presenta una comparativa final de los modelos; se emplean cuatro criterios fundamentales para evaluar la calidad de los modelos utilizados en la tarea de simplificación:

- **Calidad de la simplificación:** Evalúa la claridad del texto generado y se mira si la frase simplificada mantiene su significado original.
- **Cumplimiento de normas de lectura fácil:** Verifica si el resultado se ajusta a las reglas de lectura fácil, explicadas en la sección 3.1.1.

- **Restricciones de uso:** Se consideran todas las limitaciones del uso de la API, este criterio tiene mayor relevancia respecto al resto, ya que es fundamental hacer todas las llamadas necesarias para conseguir el resultado esperado.
- **Alucinaciones del modelo:** Mide la tendencia del modelo a inventar información que no aparece en el texto original, esto es especialmente problemático en la tarea de simplificación y en la de extracción de entidades y relaciones.

Según estos criterios, el modelo de Groq basado en *LLaMA 4* con salida estructurada en formato JSON proporciona los mejores resultados. Ofrece una alta calidad de simplificación, se ajusta a las normas de lectura fácil, la API de Groq proporciona llamadas casi ilimitadas en el plan gratuito, y no tiene alucinaciones. Todo esto lo convierte en la mejor opción para la tarea de simplificación y extracción de entidades y relaciones, en comparación con otras alternativas como Cohere (aunque eficaz, su API está mucho más limitada) o Salamandra (menos flexible y capaz que Groq en términos de salida y velocidad de respuesta).

3.2. Extracción de entidades y relaciones semántica

Una vez realizada la simplificación del texto, procedemos a la siguiente etapa, extracción de entidades y relaciones. En esta etapa, a partir de un texto simplificado, se extraen las entidades clave con sus relaciones. Se implementan dos modalidades, enfoques clásicos y aplicación de LLM para la extracción de entidades.

3.2.1. Enfoques clásicos y sus limitaciones

En esta primera etapa se implementaron modelos ya existentes capaces de reconocer entidades y relaciones de un texto, se usaron: spaCy, Stanza, Textrazor y modelos de transformers como *bert-spanish-cased-finetuned-ner*.

3.2.1.1. spaCy y Stanza

Estos modelos funcionan de manera muy similar y producen los mismos resultados. En primer lugar, se carga un modelo pre-entrenado; posteriormente, se hace una llamada con el texto simplificado y, finalmente, se extraen de manera programática las entidades y relaciones. Una de las principales ventajas de este modelo es su velocidad de respuesta; al tener ambos modelos descargados localmente, las llamadas tardan milisegundos en ejecutarse.

3.2.1.2. Textrazor

A diferencia de los modelos descritos en la sección 3.2.1.1, Textrazor es un modelo en la nube el cual permite ser llamado mediante APIs; es un modelo más complejo y está mejor entrenado en comparación con los anteriores, sección 3.2.1.1. Este es capaz de proporcionar, por cada entidad extraída, el nivel de importancia de esa entidad sobre la frase, además de proporcionar la confianza sobre la entidad identificada. Estos valores se encuentran en el intervalo [0,1]. Este modelo destaca por ser capaz de identificar entidades fuera del rango Persona, Lugar, Organización. En el Listing 3.4 se muestra un ejemplo de la salida de Textrazor:

```
input: "La paella contiene arroz y marisco."
output:
ENTIDADES ENCONTRADAS:
- Marisco
  - Importancia : (0.83)
  - Confianza: [1.00]
- Arroz
  - Importancia : (0.31)
  - Confianza: [1.00]
- Paella
  - Importancia : (1.00)
  - Confianza: [1.00]
```

Listing 3.4: Salida del modelo Textrazor

3.2.1.3. Transformers (bert-spanish-cased-finetuned-ner)

Finalmente, se implementó un modelo BERT, de la librería transformers, especializado en la extracción de entidades en castellano. Este modelo, al igual que el anterior, es capaz de proporcionar una puntuación de relevancia por cada entidad encontrada; además, es capaz de identificar el tipo de entidad (Persona, Organización, Lugar, etc.). En el Listing 3.5 se muestra un ejemplo de la salida de BERT:

```
input: "Rodrigo es el presidente de la Republica Argentina. El vive en Buenos Aires."
output:
ENTIDADES ENCONTRADAS:
{'entity_group': 'PER', 'score': (0.9989635), 'word': 'Rodrigo'}
{'entity_group': 'ORG', 'score': (0.76571685), 'word': 'Republica'}
{'entity_group': 'LOC', 'score': (0.54632), 'word': 'Argentina'}
{'entity_group': 'LOC', 'score': (0.999418), 'word': 'Buenos Aires'}
```

Listing 3.5: Salida del modelo BERT

3.2.1.4. Limitaciones de los modelos clásicos

Los enfoques clásicos para extraer entidades y relaciones del texto tienen bastantes limitaciones, especialmente si los comparamos con los modelos de lenguaje actuales. Uno de los principales problemas es que estos modelos están pensados para reconocer solo unas categorías de entidades, como personas, lugares u organizaciones. Esto únicamente es útil para tareas básicas, pero no lo es si necesitamos detectar cosas más específicas, como ingredientes, objetos u otro tipo de entidades concretas. En el caso de este Trabajo de Fin de Grado no bastaría con identificar conceptos generales como nombres de personas u organizaciones, sino que necesitamos que sea capaz de identificar todas las categorías posibles. Además, estos modelos no son capaces de entender correctamente las relaciones entre entidades.

En el caso de que se optase por implementar únicamente modelos clásicos, habría que entrenarlos individualmente para que sean capaces de encontrar las entidades y relaciones que queremos. Esto supone un gran esfuerzo, ya que habría que entrenarlos para cada una de las categorías que queramos identificar.

Por todo esto, se hace necesario buscar soluciones más modernas, como los modelos de lenguaje grandes, que veremos en el siguiente apartado.

3.2.2. Aplicación de LLM para extracción de entidades y relaciones

Para extraer entidades y relaciones del texto, se ha optado por utilizar modelos de lenguaje, concretamente el modelo LLaMA 4 disponible a través de la plataforma Groq. Este modelo no necesita un entrenamiento o definir reglas complejas, basta con proporcionarle un buen prompt y el modelo responde con el resultado esperado. Se definieron dos funciones, una para identificar las entidades y otra para identificar las relaciones.

3.2.2.1. Extracción de entidades

La primera tarea consiste en detectar las entidades que aparecen en el texto proporcionado como entrada. Para ello, se utiliza un modelo al que se le indica, mediante un prompt de sistema, que actúe como un experto en procesamiento de lenguaje natural. A continuación, se le proporciona un prompt específico en el que se le instruye para identificar únicamente aquellas entidades que estén mencionadas de forma explícita y literal en el texto, evitando cualquier tipo de interpretación o razonamiento adicional. Además, se le pide que devuelva también los atributos directamente asociados a cada entidad, como pueden ser cantidades, unidades o descripciones. En la llamada del modelo se establece el formato de salida que debe proporcionar el modelo; se ha elegido una estructura en formato JSON para una mayor facilidad a la hora de tratar los datos ya procesados. Por ejemplo, un texto como:

> *La paella de marisco tiene 200 gramos de atún, 300 gramos de gambas y 450 gramos de arroz.*

El modelo genera una salida estructurada en formato JSON que se muestra en el Listing 3.6:

```
{
  "entidades": [
    {
      "nombre": "atun",
      "tipo": "ingrediente",
      "atributos": {
        "cantidad": "200",
        "unidad": "gramos"
      }
    }...
    {
      "nombre": "paella de marisco",
      "tipo": "plato",
      "atributos": {}
    }
  ]
}
```

Listing 3.6: Formato salida de las entidades

3.2.2.2. Extracción de relaciones

Una vez identificadas las entidades explícitas presentes en el texto, el siguiente paso consiste en determinar las relaciones existentes entre ellas. Para esto, se utiliza el mismo modelo LLaMA 4 (con diferente API para no pasar del límite gratuito permitido), al que también se le proporciona un prompt preciso.

En este caso, el prompt le indica al modelo que debe actuar como experto en extracción de relaciones semánticas, y se le proporciona tanto el texto original como la lista de entidades extraídas en el paso anterior, sección 3.2.2.1. El prompt le instruye al modelo que se identifiquen únicamente relaciones, que estén de forma clara y explícita en el texto, sin deducir ni interpretar conexiones implícitas.

Además, se le especifica que no debe generar relaciones duplicadas ni redundantes, y que para cada

par de entidades relacionadas, escoja la dirección más directa de la relación. Si alguna relación incluye atributos como cantidades o unidades (por ejemplo, «200 gramos»), también se le pide que los asocie, pero únicamente si están vinculados en la frase original; de esta manera evitamos que el modelo alucine e interprete relaciones inexistentes.

El resultado devuelto por el modelo sigue la salida estructurada en formato JSON que se muestra en el Listing 3.7:

```
{
    "relaciones": [
    {
        "entidad1": "paella de marisco",
        "relacion": "contiene",
        "entidad2": "atun",
        "atributos": {
            "cantidad": "200",
            "unidad": "gramos"
        }
    }
  ]
}
```

Listing 3.7: Formato salida de las relaciones

Este formato facilita el uso de las relaciones obtenidas, ya que pueden ser directamente empleadas para su visualización en un grafo sin necesidad de ser procesadas. Al igual que en la extracción de entidades, todo el proceso es muy rápido y flexible, gracias a la capacidad del modelo para devolver resultados directamente estructurados, sin necesidad de convertirlos manualmente.

3.2.2.3. Ventajas del enfoque con LLM

El uso de LLM ofrece varias ventajas frente a los métodos clásicos:

- No requiere entrenamiento previo ni reglas manuales, basta con definir un buen prompt.
- Identifica no solo entidades comunes (persona, lugar, organización), sino cualquier tipo de identidad sin importar el contexto.
- Devuelve salidas estructuradas en JSON, lo que facilita su integración en otros módulos, como la visualización gráfica.
- El tiempo de respuesta de Groq es extremadamente rápido, con una media de seiscientos milisegundos por respuesta.

El uso de modelos de lenguaje permite automatizar el análisis semántico del texto con alta precisión, facilitando el uso de sus respuestas para otros módulos, como la representación gráfica.

3.3. Implementación del modelo Groq

Una vez analizados los diferentes modelos, se llega a la conclusión de implementar los modelos de lenguaje de la plataforma Groq, tanto para la adaptación del texto a lectura fácil como para la extracción de entidades y relaciones clave. Tal y como se menciona en la sección 3.1.2, las principales ventajas de Groq son: la velocidad de respuesta de los modelos, el acceso a modelos de última generación y, lo más importante, el acceso gratuito a todos sus modelos.

La solución final consiste en un pipeline, donde cada función hace una tarea específica y le pasa el resultado a la siguiente. Estas tareas siguen el siguiente orden:

1. Adaptar el texto a lectura fácil por etapas, siguiendo las normas de la UNE.
2. Detectar las entidades que aparecen en el texto.
3. Detectar las relaciones entre las entidades ya encontradas.

A continuación, se explica en detalle cuál es la lógica detrás de cada una de estas tareas.

3.3.1. Simplificación del texto por etapas

El primer paso consiste en tomar un texto que puede contener tecnicismos, frases largas, expresiones difíciles de comprender, y transformarlo en un texto adaptado a la lectura simple, es decir, más sencillo y accesible. Para conseguirlo, se diseña un proceso que hace la simplificación paso a paso; esto se implementa así para obtener los mejores resultados en la simplificación. A partir de las normas explicadas en la sección 3.1.1, se detallan las instrucciones del modelo para el cumplimiento de cada etapa:

- **Reescritura en voz afirmativa:** El modelo transforma oraciones negativas a afirmativas, siempre que esto no cambie su significado. No modifica frases que ya están en voz afirmativa, y evita usar negaciones dobles innecesarias.
- **División en frases cortas:** Esta etapa divide cualquier frase con más de una idea principal en frases separadas y simples. También divide expresiones en las que un sustantivo va acompañado de una explicación inmediata. No reescribe, no elimina ni cambia el orden original de las palabras, simplemente separa enunciados cuando es necesario.
- **Simplificación léxica:** El modelo sustituye palabras cultas, técnicas o difíciles por otras más sencillas. Si una palabra técnica (como un término médico) es esencial para entender el texto, se mantiene y se añade una nueva frase con su explicación literal. No se usan paréntesis ni comas explicativas.
- **Formato accesible para números:** Se sustituyen números difíciles o poco comunes por cifras comprensibles. Los números muy grandes pueden convertirse en comparaciones. No se alteran fechas ni medidas exactas. Si los números ya son adecuados, se mantienen.
- **Eliminar abreviaturas y siglas:** En este paso se reemplazan abreviaturas como «gr.» o siglas poco conocidas por sus equivalentes completos, como «gramos» o «por ejemplo». El modelo no cambia palabras normales, ni explica ni resume expresiones.
- **Corrección ortográfica:** Por último, el modelo corrige tildes, espacios o errores ortográficos sin cambiar ni añadir palabras nuevas. Si no hay errores, no aplica ningún cambio. Esta etapa no corresponde a ninguna norma de la UNE, pero es necesaria para obtener una frase final comprensible.

Cada una de estas etapas se envía por separado al modelo como un prompt; además, se le indica al modelo aplicar como formato de salida un JSON. De esta manera, se obliga al modelo a seguir una estructura de salida. Este formato JSON tiene la estructura mostrada en el Listing 3.8:

```
1     {
2   "simplified_text": "La paella de marisco tiene doscientos gramos de atun.",
3   "reasoning": "Se ha cambiado '200gr.' por 'doscientos gramos'.",
4   "input_text": "La paella de marisco tiene 200gr. de atun.",
5   "applied_rules": ["Reemplazar abreviaturas y numeros grandes"]
6     }
```

Listing 3.8: Estructura de salida del JSON

Gracias a esta estructura, se puede guardar toda la información de cada paso: cómo era el texto antes, qué cambio se hizo, por qué, y qué reglas se aplicaron. Así, se pueden observar los cambios que ha hecho el modelo por cada etapa y el porqué, lo que nos permite afinar los prompts en caso de error o un resultado inesperado.

3.3.2. Extracción de entidades

Una vez tenemos la versión simplificada del texto, el siguiente paso consiste en identificar las entidades relevantes. Estas entidades son necesarias para posteriormente hacer una representación visual de la frase.

Para lograrlo, el sistema hace una nueva llamada al modelo de Groq, pero ahora con un nuevo tipo de instrucciones. En este caso, se le aplica un prompt de sistema, esto es una frase que le dice al modelo cómo debe actuar, como por ejemplo: «Eres un experto en procesamiento de lenguaje». Y se le aplica otro prompt más que le indique que extraiga únicamente las entidades que aparecen de forma clara y literal en el texto, sin suponer nada.

Por cada entidad detectada, el modelo devuelve un JSON con la estructura mostrada en el Listing 3.9:

```
{
  "entidades": [
    {
      "nombre": "atun",
      "tipo": "ingrediente",
      "atributos": {
        "cantidad": "200 gramos"
      }
    },
    {
      "nombre": "paella de marisco",
      "tipo": "plato",
      "atributos": {}
    }
  ]
}
```

Listing 3.9: Salida esperada por entidad

Esto permite almacenar en una JSON todos los elementos importantes del texto, listos para ser procesados en el siguiente paso.

3.3.3. Extracción de relaciones

Una vez extraídas las entidades, el último paso es detectar qué relación hay entre ellas. Por ejemplo, si el texto dice que «La paella contiene 200 gramos de atún», el sistema debe entenderlo como «La paella contiene atún». Además, el modelo recoge de la lista anterior las entidades (con sus atributos) y el texto original, y se lo pasa de nuevo al modelo con instrucciones muy específicas: estas instrucciones le indican al modelo convertir las entidades y los atributos ya extraídos en relaciones.

Por ejemplo, en el Listing 3.10 podemos observar cómo, a partir de la frase *La paella de marisco tiene 200 gramos de atún, 300 gramos de gambas y 450 gramos de arroz.* el atributo cantidad: 200 gramos de atún se transforma en una relación.

```
{
  "relaciones": [
    {
      "entidad1": "paella de marisco",
      "relacion": "contiene",
      "entidad2": "atun",
      "atributos": {
        "cantidad": "200",
        "unidad": "gramos"
      }
    }
  ]
}
```

Listing 3.10: Ejemplo de salida esperada por relación

Estas relaciones, una vez procesadas y extraídas de su respectivo JSON, están listas para ser mostradas en una representación visual donde cada entidad es un nodo (un círculo), y las relaciones son líneas que las conectan, con etiquetas que indican la acción como, por ejemplo, «contiene».

3.4. Representación gráfica del texto simplificado

Una parte esencial de este Trabajo de Fin de Grado es la capacidad de generar una representación visual del texto una vez que ha sido simplificado y analizado para extraer sus entidades y relaciones. Esta representación gráfica permite ver todas las entidades clave del texto y cómo se relacionan entre ellas, haciendo la información más accesible y fácil de entender.

3.4.1. Diseño de grafos y mapas conceptuales

Una vez que se han extraído las entidades y las relaciones entre ellas, esos datos se transforman en un grafo visual. Para lograrlo, se sigue el siguiente proceso:

1. **Extracción y procesamiento de los datos:** A partir de las respuestas JSON que devuelve el modelo Groq tras los pasos de simplificación, extracción de entidades y relaciones, se crean dos tablas con ayuda de la librería *pandas*:

    - Una tabla para las entidades con sus respectivos nombres, tipos y atributos, véase figura 3.1.

	nombre	tipo	atributos
0	paella valencia	plato	{}
1	pollo	ingrediente	{'cantidad': '300 gramos'}
2	arroz	ingrediente	{'cantidad': '350 gramos'}

Figura 3.1: Ejemplo de tabla con entidades extraídas del texto

    - Una tabla para las relaciones, con las entidades conectadas y el tipo de relación, véase figura 3.2.

	entidad1	relacion	entidad2
0	paella valencia	contiene	pollo
1	paella valencia	contiene	arroz
2	pollo	tiene cantidad	300 gramos
3	arroz	tiene cantidad	350 gramos

Figura 3.2: Ejemplo de tabla con relaciones entre entidades

2. **Conversión a nodos y conexiones:** Con estos datos, se utiliza la librería *Pyvis* para crear un grafo visual interactivo. El grafo se genera dinámicamente en HTML, este grafo es exportado, en pasos posteriores, para su uso en la página web. En este grafo cada entidad es un nodo, con un color distinto según el tipo de la identidad y las relaciones son flechas que conectan entidades.

3. **Procesamiento de los atributos:** Cuando una entidad incluye atributos (por ejemplo, «300 gramos»), ese valor se convierte también en un nodo, de este modo se mantiene todo el significado de la frase en el grafo. Estos nodos tienen un diseño diferente (forma de caja) para distinguirlos de las entidades principales.

3.4.2. Criterios de visualización

Para que la representación visual no sea confusa, se han seguido varios criterios de diseño:

- **Colores por tipo de entidad:** Cada tipo de entidad (ingrediente, lugar, persona, acción, etc.) tiene un color diferente, lo que ayuda a identificarlas fácilmente.

- **Formas:** Los nodos principales tienen forma de círculo, mientras que los atributos se muestran en forma de caja.

- **Diseño interactivo:** El grafo es completamente iterativo; se puede mover, hacer zoom y reorganizar cada uno de los nodos y relaciones como el usuario desee. Mejorando así la visualización.

- **Algoritmo de distribución de nodos:** Se ha empleado un algoritmo de fuerzas (force-directed layout) para distribuir automáticamente los nodos y que, al haber muchas relaciones, no se solapen entre sí. Esto mejora la presentación y visualización del grafo.

A continuación, se muestra un ejemplo generado por el sistema a partir del siguiente texto:

> *La paella valenciana tiene 300 gramos de pollo y 350 gramos de arroz.*

El sistema extrae las entidades principales: paella valenciana, arroz, pollo. Además, identifica las relaciones «contiene» entre la paella y cada ingrediente, junto con sus respectivas cantidades, realizando el grafo, véase figura 3.3.

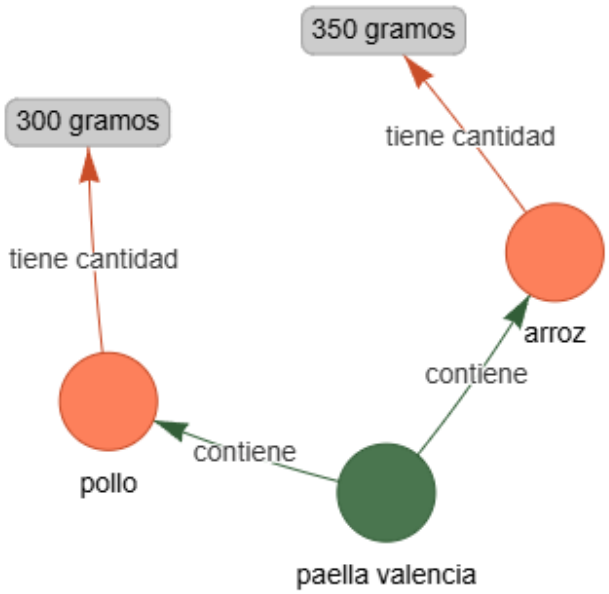


Figura 3.3: Representación gráfica del texto simplificado mostrando entidades y relaciones extraídas

3.5. Desarrollo de la aplicación web

Para finalizar el proyecto, se desarrolló una aplicación web que permite introducir texto, simplificando siguiendo las normas de lectura fácil, extraer entidades y relaciones clave y visualizar los resultados con un grafo interactivo. Este desarrollo se ha dividido en dos fases:

1. Una versión centrada en la validación de resultados, para ver el progreso de cada etapa.
2. Una versión final orientada al usuario, con un diseño más visual y accesible.

3.5.1. Arquitectura y tecnologías empleadas

La aplicación está desarrollada en Python empleando Streamlit, una herramienta que permite diseñar interfaces web de manera muy rápida y sin conocimientos previos de lenguajes como HTML o CSS. Se eligió esta herramienta no solo por su gran compatibilidad con la librería Pyvis, empleada para crear el grafo visual, sino también por su capacidad para crear objetos HTML interactivos y, lo más importante, tiene la capacidad de poder desplegar la página web en la nube, sin necesidad de tenerla en local.

3.5.1.1. Primera fase: Interfaz de prueba y retroalimentación

Esta primera fase cuenta con la siguiente estructura, donde se separa la interfaz del usuario, la simplificación y extracción de entidades y relaciones clave, y la generación del grafo. A continuación, en la figura 3.4, se muestra el árbol de carpetas y una breve explicación del contenido de cada una:

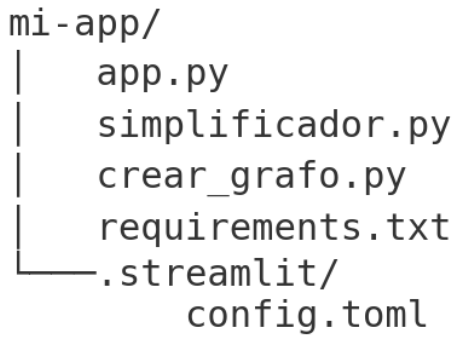


Figura 3.4: Estructura de carpetas del proyecto

- **app.py:** Archivo principal que ejecuta la aplicación web. Contiene el diseño de la interfaz con Streamlit, entrada de texto, botones, visualización del grafo e integración con el resto de módulos. Además este archivo es el que se emplea para desplegar la página web en la nube.

- **simplificador.py:** Archivo que contiene las funciones que simplifican texto por etapas, aplican los prompts personalizados al modelo Groq y se extraen las entidades y relaciones.

- **crear_grafo.py:** Archivo encargado de convertir las entidades y relaciones extraídas en un grafo interactivo empleando la librería Pyvis. Este grafo es el que se muestra en la interfaz web.

- **requirements.txt:** Este archivo es necesario para que el servicio de Streamlit descargue las dependencias correspondientes para el despliegue de la página web.

- **.streamlit/config.toml:** Es el archivo de configuración de Streamlit, en él se definen variables como el ancho de la página, el título del navegador, y otras opciones para personalizar el estilo de la página.

Esta primera fase está pensada como una herramienta para probar y analizar, a tiempo real, el comportamiento del sistema completo, es decir, para visualizar proceso a proceso desde la entrada de texto hasta el grafo generado. Esta herramienta es fundamental durante el desarrollo, ya que permite observar los resultados de cada etapa, verificar si el modelo funciona como se espera y detectar errores. Las secciones principales son:

- **Entrada de texto:** La aplicación muestra un área de texto donde el usuario puede introducir una frase en castellano. Primero se verifica que el campo no esté vacío, y una vez pulsado el botón «Procesar», se ejecuta todo el sistema. En la figura 3.5 se puede observar su interfaz.

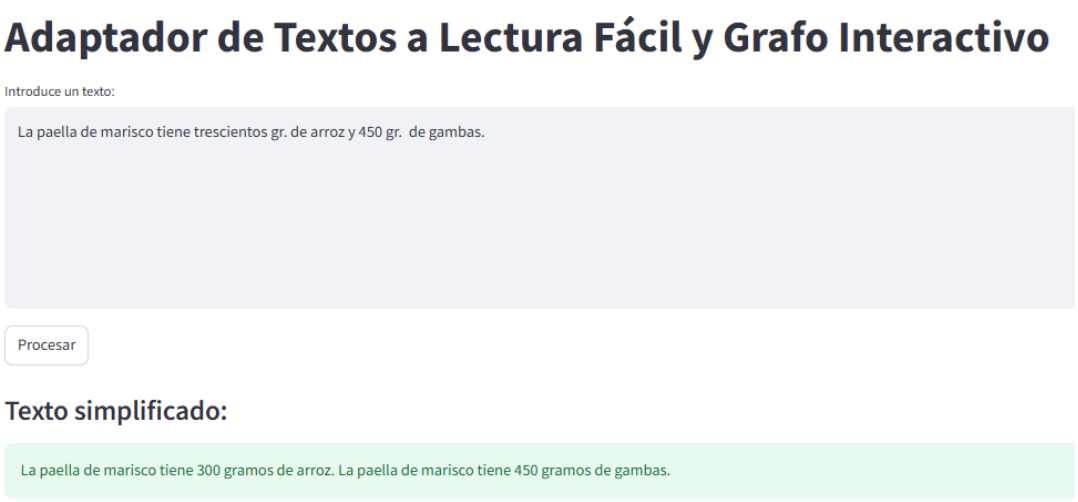


Figura 3.5: Visualización del texto original introducido y el texto simplificado

- **Simplificación paso a paso:** Se aplica un pipeline con las etapas de simplificación. Se muestra un desplegable que al pulsar permite visualizar el procesamiento que se hace para simplificar una frase. En la figura 3.6 se muestra la etapa relevante aplicada.

Figura 3.6: Visualización de las etapas y procesamiento seguido para la simplificación

- **Extracción de entidades:** Una vez simplificado el texto, se extraen automáticamente las entidades. Estas se muestran en una tabla para ver claramente los elementos detectados por el modelo, junto con su tipo, véase figura 3.7.

Nombre	Tipo	Atributos
Paella de marisco	Plato	Ingrediente principal: Arroz y Gambas
Arroz	Ingrediente	Cantidad: 300 gramos
Gambas	Ingrediente	Cantidad: 450 gramos

Figura 3.7: Tabla con entidades extraídas del texto simplificado

- **Extracción de relaciones:** Además de las entidades, el sistema identifica las relaciones semánticas entre ellas. Por ejemplo, la relación «contiene» entre un plato y sus ingredientes, o la cantidad asociada a cada ingrediente, véase figura 3.15.

Entidad 1	Relación	Entidad 2
Paella de marisco	tiene ingrediente	Arroz
Paella de marisco	tiene ingrediente	Gambas
Arroz	cantidad	300 gramos
Gambas	cantidad	450 gramos

Figura 3.8: Tabla con las relaciones extraídas entre entidades

- **Visualización del grafo interactivo:** Finalmente, se construye un grafo donde se representan todas las entidades y relaciones extraídas. En la figura 3.9, se puede ver el resultado final de todo el proceso de adaptación de un texto a lectura fácil.

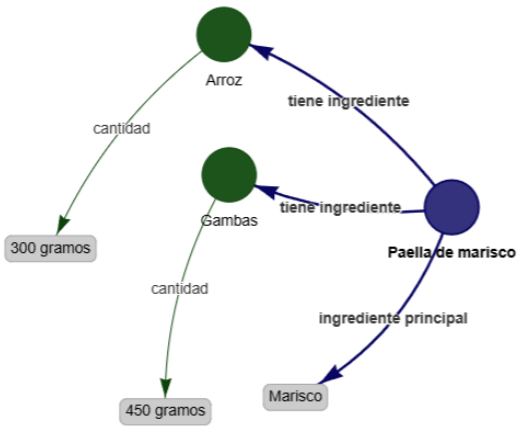


Figura 3.9: Grafo visual resultante tras aplicar todo el proceso de adaptación a lectura fácil

3.5.1.2. Segunda fase: Interfaz final para el usuario

En esta segunda fase se muestra la versión final de la página web; esta está completamente adaptada al usuario, por lo que no contiene el proceso intermedio de simplificación del texto, únicamente se muestra aquella información relevante para el usuario. Además, se introdujeron funciones adicionales para mejorar la experiencia del usuario. Entre los principales cambios destacan la reorganización de la página web y el asistente virtual.

A continuación, en la figura 3.10, se muestra la nueva estructura del proyecto y el contenido general de aquellas clases que se han modificado.

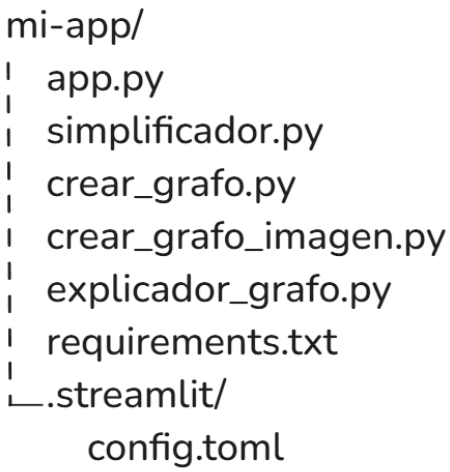


Figura 3.10: Estructura final de carpetas del proyecto

- **app.py:** Es el archivo principal que lanza la aplicación web. En esta nueva fase se ha simplificado la interfaz para que sea más clara y con menos información. Además, se le han añadido dos nuevas funcionalidades: ahora llama a `crear_grafo_imagen.py` para generar una imagen del grafo, y a `explicador_grafo.py`, que permite activar un asistente virtual para hacer preguntas sobre el contenido del grafo.
- **simplificador.py:** Se ha añadido una mejora importante: ahora el sistema genera el grafo tres veces con diferentes resultados, esto es necesario ya que los modelos de lenguaje no siempre responden igual y puede que en una iteración devuelvan la llamada correcta y en la siguiente no. Una vez obtenido una lista con los tres grafos, en formato JSON, se utiliza otro modelo para comparar esos tres resultados y elegir el mejor.
- **crear_grafo_imagen.py:** Este archivo genera una imagen del grafo, igual que el grafo interactivo, pero pensada para ser usada con el asistente virtual. Usa la librería *matplotlib* y convierte la imagen a base64 para que se pueda enviar al modelo de lenguaje.
- **explicador_grafo.py:** Aquí se implementa el asistente virtual que responde preguntas del usuario. Este asistente recibe la imagen del grafo, el texto original y simplificado, todas las entidades y relaciones, y las explicaciones de cada etapa de simplificación empleadas. Con esa información, puede responder a cualquier duda sobre el grafo o sobre cómo se ha simplificado el texto.

A continuación se muestra el diseño final de la página web:

- **Entrada de texto:** La aplicación muestra un área de texto donde el usuario puede introducir una

frase en castellano. Primero se verifica que el campo no esté vacío, y una vez pulsado el botón «Procesar», se ejecuta todo el sistema, esto no fue modificado respecto a la anterior versión.

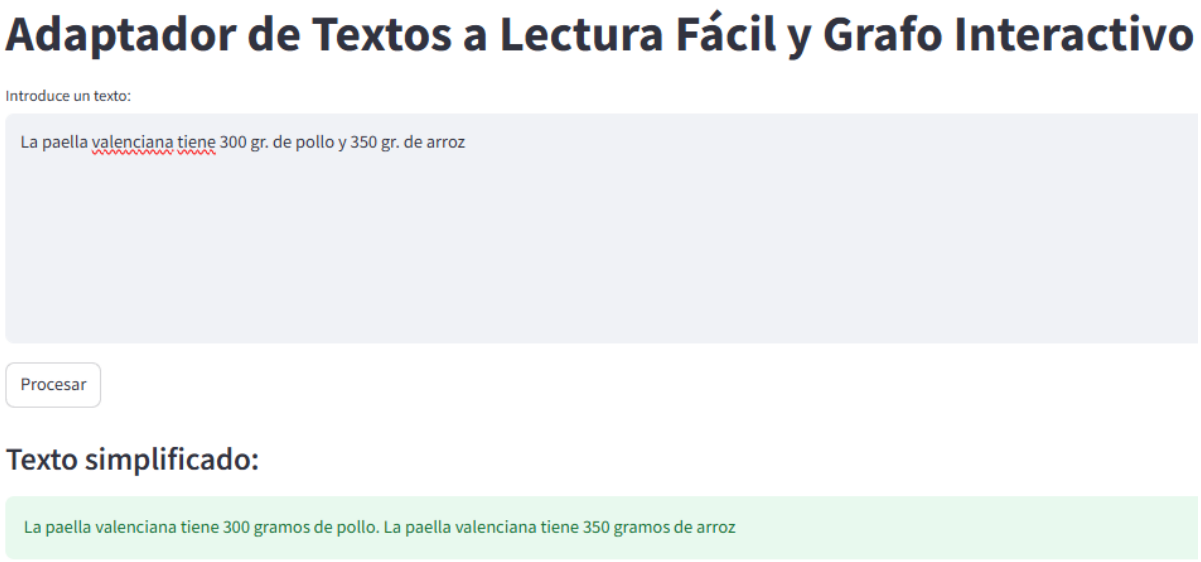


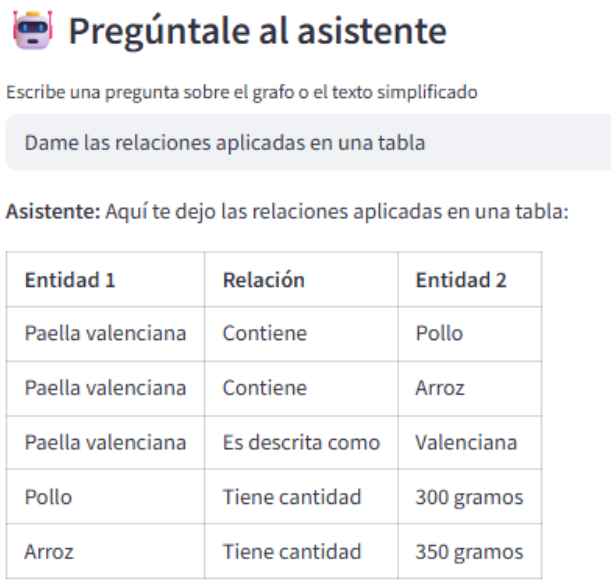
Figura 3.11: Visualización del texto original introducido y el texto simplificado

- **Extracción de entidades:** Una vez que el texto ha sido simplificado, se extraen automáticamente las entidades más relevantes. Estas entidades, a diferencia de la anterior versión, no se muestran por pantalla. No obstante, se las podemos pedir al asistente, el cual nos devolverá una tabla con aquellas aplicadas en el grafo, véase figura 3.12

Nombre	Tipo	Atributos
Paella valenciana	Plato	Cantidad: 300 gramos de pollo y 350 gramos de arroz, Descripción: valenciana
Pollo	Ingrediente	Cantidad: 300 gramos
Arroz	Ingrediente	Cantidad: 350 gramos

Figura 3.12: Tabla con entidades extraídas del asistente

- **Extracción de relaciones:** Además de las entidades, el sistema detecta las relaciones semánticas entre ellas, como «contiene», «tiene cantidad», o «proviene de». Las podemos obtener pidiéndoselas al asistente, véase figura 3.13.



Pregúntale al asistente

Escribe una pregunta sobre el grafo o el texto simplificado

Dame las relaciones aplicadas en una tabla

Asistente: Aquí te dejo las relaciones aplicadas en una tabla:

Entidad 1	Relación	Entidad 2
Paella valenciana	Contiene	Pollo
Paella valenciana	Contiene	Arroz
Paella valenciana	Es descrita como	Valenciana
Pollo	Tiene cantidad	300 gramos
Arroz	Tiene cantidad	350 gramos

Figura 3.13: Relaciones extraídas del asistente

- **Visualización del grafo interactivo:** Se genera un grafo interactivo a partir de las entidades y relaciones. El usuario puede interactuar por él directamente desde la interfaz, reorganizando sus nodos como desee. La figura 3.14 contiene el grafo final tras aplicar todas las transformaciones.

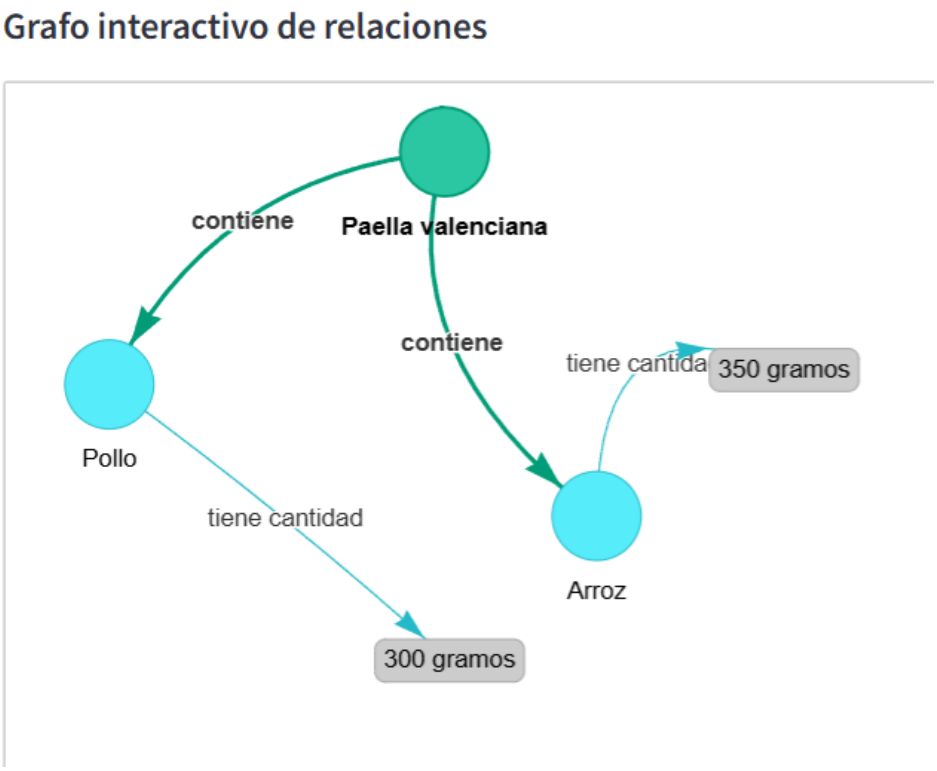


Figura 3.14: Relaciones extraídas del asistente

- **Generación de imagen del grafo:** A partir de los mismos datos, también se genera una imagen en formato PNG del grafo, esta imagen, en el formato correcto, se emplea como parte del prompt para el asistente virtual.
- **Asistente virtual:** Se incluye un chatbot capaz de responder preguntas sobre el grafo, el texto original y el proceso de adaptación. Este asistente analiza la imagen generada del grafo y toda la información de cada proceso, además, es capaz de mantener un conversación en base a otras preguntas ya realizadas, ya que conserva un registro del chat.

3.6. Pruebas del sistema desarrollado

En esta sección se presenta la evaluación del sistema con respecto a su precisión y efectividad. Para ello se pone en comparación con modelos LLM con mayores capacidades que LLaMA, entre ellos el modelo de GPT-4o. A lo largo de esta etapa se realizan seis comparaciones, tres de adaptación de texto a lectura fácil y otras tres visuales, comparando los grafos entidad-relación.

3.6.1. Pruebas de adaptación del texto a lectura

Para esta primera evaluación, se comparan los modelos en base a su capacidad para adaptar un texto a su respectiva lectura fácil, en base a las normas de la UNE. Ambos modelos reciben el mismo prompt con las etapas y restricciones para adaptar un texto a su respectiva lectura fácil, sección 3.1.1. A continuación, se muestran las tres pruebas realizadas.

3.6.1.1. Primera prueba: formato accesible de números

- **Texto original:** La paella de marisco tiene trescientos gr. de arroz y 450 gr. de gambas.
- **Texto adaptado (LLaMA):** La paella de marisco tiene 300 gramos de arroz. La paella de marisco tiene 450 gramos de gambas.
    - **Razonamiento del modelo:**
        - Se aplicó la regla de división de ideas porque la frase original contenía más de una idea principal unidas por «y».
        - Se aplicó la regla de simplificación de abreviaturas, reemplazando «gr». por «gramos» para mejorar la claridad.
        - Se aplicó la regla de uso de cifras, cambiando «trescientos» por «300» al ser una forma más común de representar números.
- **Texto adaptado (GPT-4o):** La paella de marisco tiene 300 gramos de arroz. Tiene 450 gramos de gambas.
    - **Razonamiento del modelo:**
        - Se aplicó la regla de uso de cifras para cambiar «trescientos» por «300» y así mejorar la claridad.
        - Se aplicó la regla de división de ideas, ya que la oración original contenía dos conceptos unidos por «y», lo que dificultaba su comprensión.
- **Observaciones:**
    - El modelo desarrollado (LLaMA) es capaz de aplicar correctamente las mismas transformaciones, siguiendo la normativa de la UNE, que un modelo más avanzado como GPT-4o.

- Sin embargo, GPT-4o omite la repetición de «La paella de marisco», haciendo el texto menos redundante.

3.6.1.2. Segunda prueba: oración con doble negación

- **Texto original:** Manuel no puede ir al concierto ni se lo puede permitir.
- **Texto adaptado (LLaMA):** Manuel no puede ir al concierto. No tiene dinero suficiente.
  - **Razonamiento del modelo:**
    - Se ha transformado «ni se lo puede permitir» en «no puede permitírselo» para mantener el significado y simplificar ligeramente la estructura, aunque sigue siendo negativa.
    - La regla se aplicó parcialmente, ya que no se pudo cambiar a afirmativa sin alterar el contenido semántico.
    - Se sustituyó «no puede permitírselo» por «no tiene suficiente dinero» porque permitírselo es una expresión culta que puede resultar difícil de comprender. La nueva frase es más sencilla y clara.
- **Texto adaptado (GPT-4o):** Manuel no puede ir al concierto. No se lo puede permitir.
  - **Razonamiento del modelo:**
    - Se aplicó la regla de división de ideas principales, separando dos conceptos que estaban unidos.
    - Se intentó simplificar la estructura manteniendo el significado, aunque se conservó la doble negación para no alterar el significado.
- **Observaciones:**
  - LLaMA reemplaza algunas palabras por términos más simples y mantiene el significado de la frase, no obstante el modelo decide cambiar «ni se lo puede permitir» por «no tiene dinero suficiente», esto, a pesar de que sea parcialmente correcto, puede llevar a errores o inconsistencias.
  - GPT-4o respeta la estructura original y divide en frases.

3.6.1.3. Tercera prueba: frase técnica

- **Texto original:** El sistema implementa técnicas de aprendizaje automático para segmentar datos no estructurados.
- **Texto adaptado (LLaMA):** El sistema usa técnicas de aprendizaje automático para separar datos que no están ordenados. El aprendizaje automático es una forma de enseñar a los ordenadores a aprender de los datos.
  - **Razonamiento del modelo:**
    - Se sustituyó «implementa» por «usa» para simplificar el vocabulario.
    - Se cambió «segmentar» por «separar» para facilitar la comprensión.
    - Se añadió una oración adicional para explicar «aprendizaje automático», ya que es un término técnico relevante para la comprensión del mensaje.

- **Texto adaptado (GPT-4o):** El sistema usa técnicas de aprendizaje automático. Sirven para segmentar datos no estructurados.
  - **Razonamiento del modelo:**
    - Se reemplazó «implementa» por «usa» para mejorar la comprensión léxica.
    - Se dividió la oración original en dos frases simples para separar la acción del propósito, facilitando su lectura.
- **Observaciones:**
  - LLaMA introduce una explicación del concepto más técnico, cumpliendo así con la norma UNE, aunque añade longitud al texto.
  - GPT-4o simplifica correctamente la estructura y el léxico, pero mantiene el término técnico «aprendizaje automático» sin explicación adicional.

3.6.2. Pruebas visuales, grafo entidad-relación :

En esta sección se prueba la capacidad de extraer entidades y relaciones de ambos modelos. Esta evaluación se realiza comparando los grafos resultantes de la extracción de entidades y relaciones. Al igual que en la comparación anterior, sección 3.6.1, ambos modelos tienen el mismo prompt para la extracción de entidades y relaciones.

3.6.2.1. Primera prueba: formato accesible de números

- **Texto adaptado:** La paella de marisco tiene 300 gramos de arroz. La paella de marisco tiene 450 gramos de gambas.
- **Grafo visual (LLaMA):**

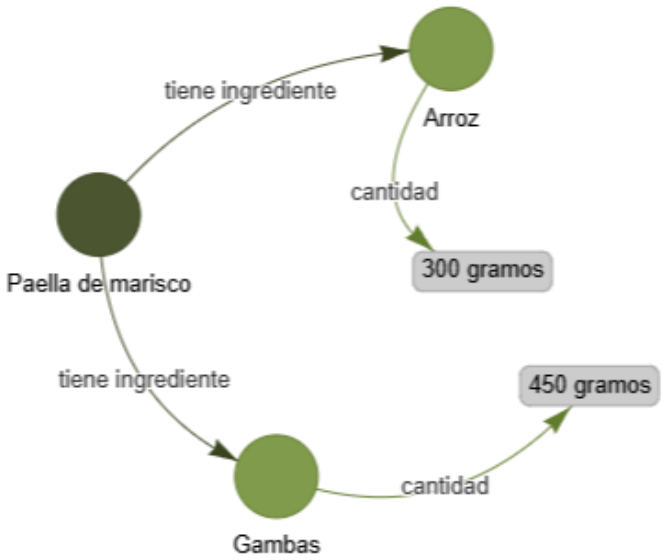


Figura 3.15: Grafo visual de la prueba 1 realizado por LLaMA

- **Texto adaptado:** La paella de marisco tiene 300 gramos de arroz. Tiene 450 gramos de gambas.
- **Grafo visual (GPT-4o):**

Figura 3.16: Grafo visual de la prueba 1 realizado por GPT

- **Observaciones:** El grafo de LLaMA muestra mejor la relación entre los ingredientes y sus cantidades por separado. El de GPT-4o es más simple visualmente, pero pierde algo de claridad por las conexiones de sus nodos.

3.6.2.2. Segunda prueba: oración con doble negación

- **Texto adaptado:** Manuel no puede ir al concierto. No tiene dinero suficiente.
- **Grafo visual (LLaMA):**

Figura 3.17: Grafo visual de la prueba 2 realizado por LLaMA

- **Texto adaptado:** Manuel no puede ir al concierto. No se lo puede permitir.
- **Grafo visual (GPT-4o):**

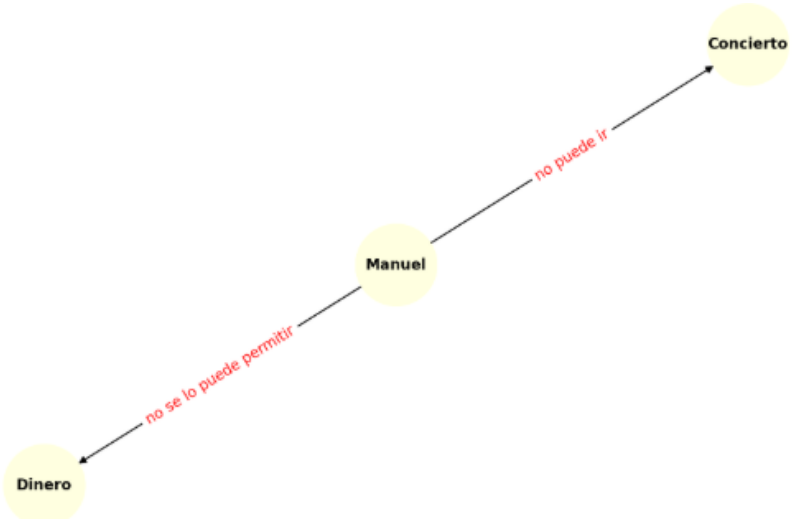


Figura 3.18: Grafo visual de la prueba 2 realizado por GPT

- **Observaciones:** El grafo de LLaMA agrupa mejor las ideas separando claramente los conceptos de «dinero» y «concierto». El resultado de GPT-4o es menos claro al representar las causas. A pesar de que el modelo de GPT-4o no relacionase «no se lo puede permitir» con «dinero», en el grafo visual si que lo hace.

3.6.2.3. Tercera prueba: frase técnica

- **Texto adaptado:** El sistema usa técnicas de aprendizaje automático para separar datos que no están ordenados. El aprendizaje automático es una forma de enseñar a los ordenadores a aprender de los datos.
- **Grafo visual (LLaMA):**

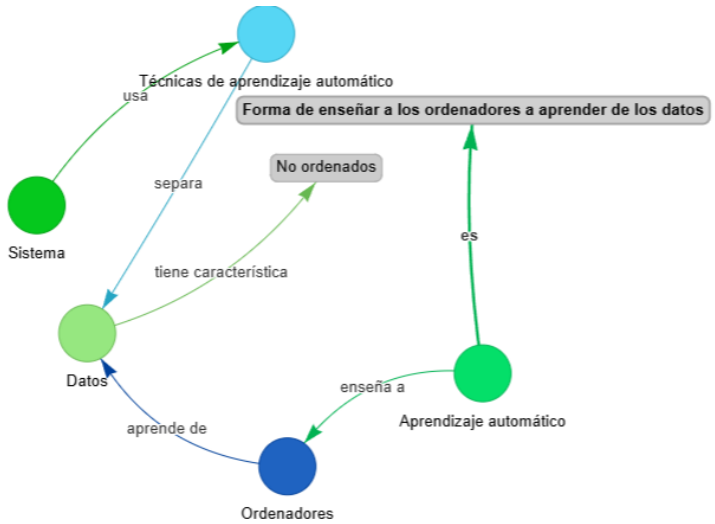


Figura 3.19: Grafo visual de la prueba 3 realizado por LLaMA

- **Texto adaptado:** El sistema usa técnicas de aprendizaje automático. Sirven para segmentar datos no estructurados.
- **Grafo visual (GPT-4o):**

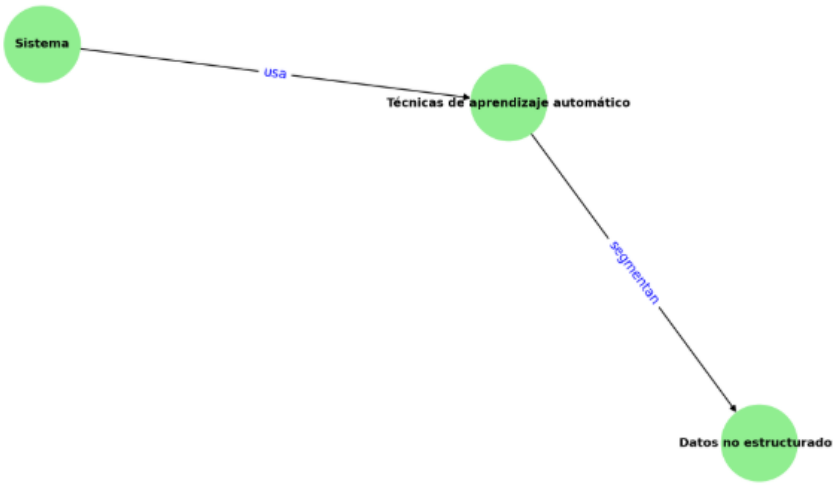


Figura 3.20: Grafo visual de la prueba 3 realizado por GPT

- **Observaciones:** Aquí hay que tener en cuenta que la adaptación de LLaMA es mucho más densa que la de GPT, por lo que el grafo desarrollado también lo es. El grafo de LLaMA contiene cierta repetición mientras que el de GPT es más directo y simple.

Como se ha podido observar, el modelo desarrollado es capaz de sobrepasar en algunos aspectos a modelos de pago de última generación como GPT-4o. No obstante, cuando se trata de oraciones más complejas y/o con un significado más profundo, GPT-4o sobrepasa a LLaMA.

Capítulo 4

Resultados

El sistema que se ha desarrollado a lo largo de este proyecto permite transformar un texto en castellano en una versión adaptada siguiendo las normas de adaptación de un texto a lectura fácil, y posteriormente representar esa adaptación de manera gráfica en una aplicación web. Este sistema permite que las personas con dificultades a la hora de leer o entender lo leído sean capaces de comprender mejor los textos.

La adaptación del texto sigue una serie de normas, estas normas incluyen pautas como el uso de frases cortas, la sustitución o explicación de términos técnicos, la escritura en voz activa y la conversión de cifras grandes o abreviaturas en expresiones más simples; estas normas están explicadas más en detalle en la sección 3.1.1

El resultado obtenido tras la simplificación es un texto más claro, adaptado a las normas de lectura fácil y con un vocabulario comprensible para personas con dificultades lectoras. Este proceso se realiza de manera automática, sin necesidad de un control.

La representación gráfica del texto simplificado es muy importante. Con la construcción de un grafo visual, el sistema permite visualizar todas las ideas clave que contiene el texto. Esto se logra con la identificación y extracción de entidades y las relaciones entre ellas a partir del texto simplificado.

El uso de grafos facilita la comprensión del contenido a través de elementos visuales. Las entidades se representan como nodos, y las relaciones como aristas entre estos nodos. Además, los atributos específicos de las entidades, como cantidades o unidades, también se encuentran en el grafo. Estos grafos emplean colores en función del tipo de entidad, formas distintas para nodos principales y atributos, y una función de fuerza para evitar que los nodos se solapen. Además, es completamente interactivo, por lo que el usuario puede mover los nodos y relaciones como desee.

Finalmente, la aplicación cuenta con un asistente virtual capaz de responder cualquier duda relacionada con el grafo resultante o con la simplificación de la frase a lectura fácil.

En resumen, los resultados obtenidos demuestran que es posible construir una aplicación automática capaz de adaptar y representar gráficamente un texto en castellano, siguiendo las normas de adaptación de textos a lectura fácil.

Capítulo 5

Conclusiones y líneas futuras

En esta sección se comentan las conclusiones obtenidas tras finalizar el desarrollo del proyecto, se explican las principales limitaciones del sistema y se proponen posibles líneas de trabajo futuras para ampliar o mejorar el sistema construido.

5.1. Conclusiones

Este proyecto me ha permitido aplicar de forma práctica muchos de los conocimientos que he ido adquiriendo a lo largo de la carrera, pero también me ha llevado a investigar y aprender cosas nuevas por mi cuenta. He investigado temas relacionados con el procesamiento de lenguaje natural, visualización de grafos, APIs, creación de páginas web en la nube con Streamlit, prompt engineering y uso de modelos de lenguaje.

También he aprendido a resolver retos difíciles, uno de los mayores ha sido controlar, lo máximo posible, lo que hacen los modelos de lenguaje: a veces devuelven la salida esperada, y en la siguiente llamada alucinan o devuelven cosas erróneas. Por eso, se implementó el diseño por etapas y el formato estructurado en JSON para tener el máximo control. También he podido desarrollar funciones adicionales, como el asistente virtual, que no se especificaban en la descripción del proyecto. Consiguiendo así otra funcionalidad más para facilitar la interacción con el usuario.

5.2. Limitaciones del proyecto

Aunque el sistema funciona correctamente en la mayoría de los casos, tiene varias limitaciones importantes.

La principal es el uso de un modelo gratuito como LLaMA, que no siempre genera los resultados esperados o precisos, sobre todo en la parte de extracción de entidades y relaciones, y como consecuencia en el desarrollo del grafo. A veces deja fuera relaciones importantes, conecta mal las entidades o las genera fuera de lugar. Además, este tipo de modelo se satura fácilmente: si se le dan prompts largos o con muchas instrucciones, ignora parte de ellas o no hace nada de lo que se le pide.

Para mejorar esto, se añadió una función que genera tres grafos distintos en lugar de uno. Después, un modelo más potente (como GPT-4-turbo) analiza los tres y elige el mejor. Aun así, el grafo final sigue estando limitado por lo que es capaz de generar LLaMA en esas tres iteraciones, por lo que los errores iniciales pueden seguir presentes. También hay veces que el modelo simplifica demasiado y

pierde información importante, o al revés, añade explicaciones innecesarias y hace el texto más largo de lo necesario.

5.3. Trabajo futuro

Además de considerar el uso de modelos más avanzados, arreglando así cualquier tipo de error en la simplificación y extracción de entidades y relaciones, hay varias líneas de mejora que podrían aplicarse en este sistema:

- **Ampliar el texto soportado:** Actualmente el sistema funciona bien con frases cortas. Sin embargo, en textos largos o complejos, el modelo suele cometer errores. Una posible mejora es adaptar el sistema para que sea capaz de dividir ese texto largo en diferentes frases y posteriormente unir los resultados, o directamente emplear modelos de mayor capacidad que puedan procesar más información de forma precisa.
- **Retroalimentación del usuario:** Una mejora a tener en cuenta consiste en permitir que el usuario pueda editar el grafo generado desde la propia interfaz, o dejar comentarios que se registren como ejemplos en el prompt para seguir mejorando los resultados.
- **Optimización de la página web:** Aunque la aplicación ya está disponible públicamente, se puede mejorar su diseño y añadir secciones más visuales, explicaciones o ejemplos guiados para facilitar su uso. Además, se pueden utilizar otras herramientas para el despliegue de esta, teniendo un mayor control y personalización.
- **Uso del asistente para continuar generando el grafo:** Una posible mejora consiste en permitir que el asistente virtual no solo responda dudas, sino que también tenga la capacidad de generar o completar automáticamente partes del grafo a partir de los comentarios de los usuarios en la conversación.

Bibliografía

[1] Olga Carreras Montoto. *Lectura Fácil: Pautas y recomendaciones. UNE 153101:2018 EX*. [En línea]. Disponible: https://olgacarreras.blogspot.com/2019/02/lectura-facil-pautas-y-recomendaciones.html. 2019.

[2] Fundación Pasqual Maragall. *La lectura fácil como herramienta de accesibilidad cognitiva para personas con Alzheimer*. [En línea]. Disponible: https://blog.fpmaragall.org/lectura-facil. 2023.

[3] Inclusión Europa. *Información para todos: Pautas europeas de la lectura fácil*. [En línea]. Disponible: https://www.plenainclusion.org/publicaciones/buscador/informacion-para-todos-pautas-europeas-de-la-lectura-facil/. 2016.

[4] Comité técnico CTN 153. *UNE 153101:2018 EX. Lectura fácil: Pautas y recomendaciones para la elaboración de documentos*. [En línea]. Disponible: https://www.une.org/encuentra-tu-norma/busca-tu-norma/norma/?c=N0060036. 2018.

[5] Sanja Štajner. «Automatic Text Simplification for Social Good: Progress and Challenges». En: *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*. [En línea]. Disponible: https://aclanthology.org/2021.findings-acl.233.pdf. Association for Computational Linguistics, 2021, págs. 2637-2652.

[6] Horacio Saggion. *Text Simplification in Simplext: Making Text More Accessible*. [En línea]. Disponible: https://www.academia.edu/57420640/Text_Simplification_in_Simplext_Making_Text_More_Accessible. 2011.

[7] Gianluca Centulani. *Understanding Hallucination in LLMs: Causes, Consequences, and Mitigation Strategies*. [En línea]. Disponible: https://medium.com/@gcentulani/understanding-hallucination-in-llms-causes-consequences-and-mitigation-strategies-b5e1d0268069. Accedido el 17 de abril de 2025.

[8] Stefan Bott y Horacio Saggion. «Automatic Simplification of Spanish Text for e-Accessibility». En: *Computers Helping People with Special Needs (ICCHP 2012)*. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 2012, págs. 527-534. DOI: 10.1007/978-3-642-31522-0_79.

[9] Cloudflare. *¿Qué son los modelos LLM? | Grandes modelos de lenguaje*. [En línea]. Disponible: https://www.cloudflare.com/es-es/learning/ai/what-is-large-language-model/. Accedido el 22 de mayo de 2025.

[10] Cohere Inc. *Introducing Aya: The World's Most Advanced Open-Weight Multilingual Model*. [En línea]. Disponible: https://cohere.com. Accedido en abril de 2025.

[11] Barcelona Supercomputing Center. *Salamandra: Multilingual Large Language Models*. [En línea]. Disponible: https://huggingface.co/BSC-LT/salamandra-7b. 2024.

[12] OpenAI. *GPT-4 Technical Report*. [En línea]. Disponible: https://openai.com/research/gpt-4. Accedido en abril de 2025.

[13] Groq Inc. *Introducing GroqCloud and LPU Inference*. [En línea]. Disponible: https://groq.com. Accedido en abril de 2025.

[14] Meta AI. *LLaMA 4: Advancing Open Foundation Models*. [En línea]. Disponible: https://ai.meta.com/llama. Accedido en abril de 2025.

[15] Wikipedia contributors. *Reconocimiento de entidades nombradas*. [En línea]. Disponible: https://es.wikipedia.org/w/index.php?title=Reconocimiento_de_entidades_nombradas&oldid=163416337. 2024.

[16] Jiaxin Li et al. «GPT-NER: Named Entity Recognition via Large Language Models». En: *arXiv preprint arXiv:2304.10428* (2023). [En línea]. Disponible: :https://arxiv.org/abs/2304.10428.

[17] Sebastian Ruder et al. *Relationship Extraction*. [En línea]. Disponible: https://nlpprogress.com/english/relationship_extraction.html. 2024.

[18] Explosion AI. *spaCy: Industrial-Strength Natural Language Processing in Python*. [En línea]. Disponible: https://spacy.io. 2023.

[19] Peng Qi et al. «Stanza: A Python Natural Language Processing Toolkit for Many Human Languages». En: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations* (2020). [En línea]. Disponible: https://aclanthology.org/2020.acl-demos.14, págs. 101-108.

[20] TextRazor Ltd. *TextRazor Natural Language Processing API*. [En línea]. Disponible: https://www.textrazor.com. Accedido en abril de 2025.

[21] 10Web. *MyMap.AI*. [En línea]. Disponible: https://10web.io/ai-tools/mymap-ai/. Accedido en abril de 2025.

[22] Dayanithi. *Streamlit in 2 Min*. [En línea]. Disponible: https://medium.com/data-and-beyond/streamlit-d357935b9c. 2023.

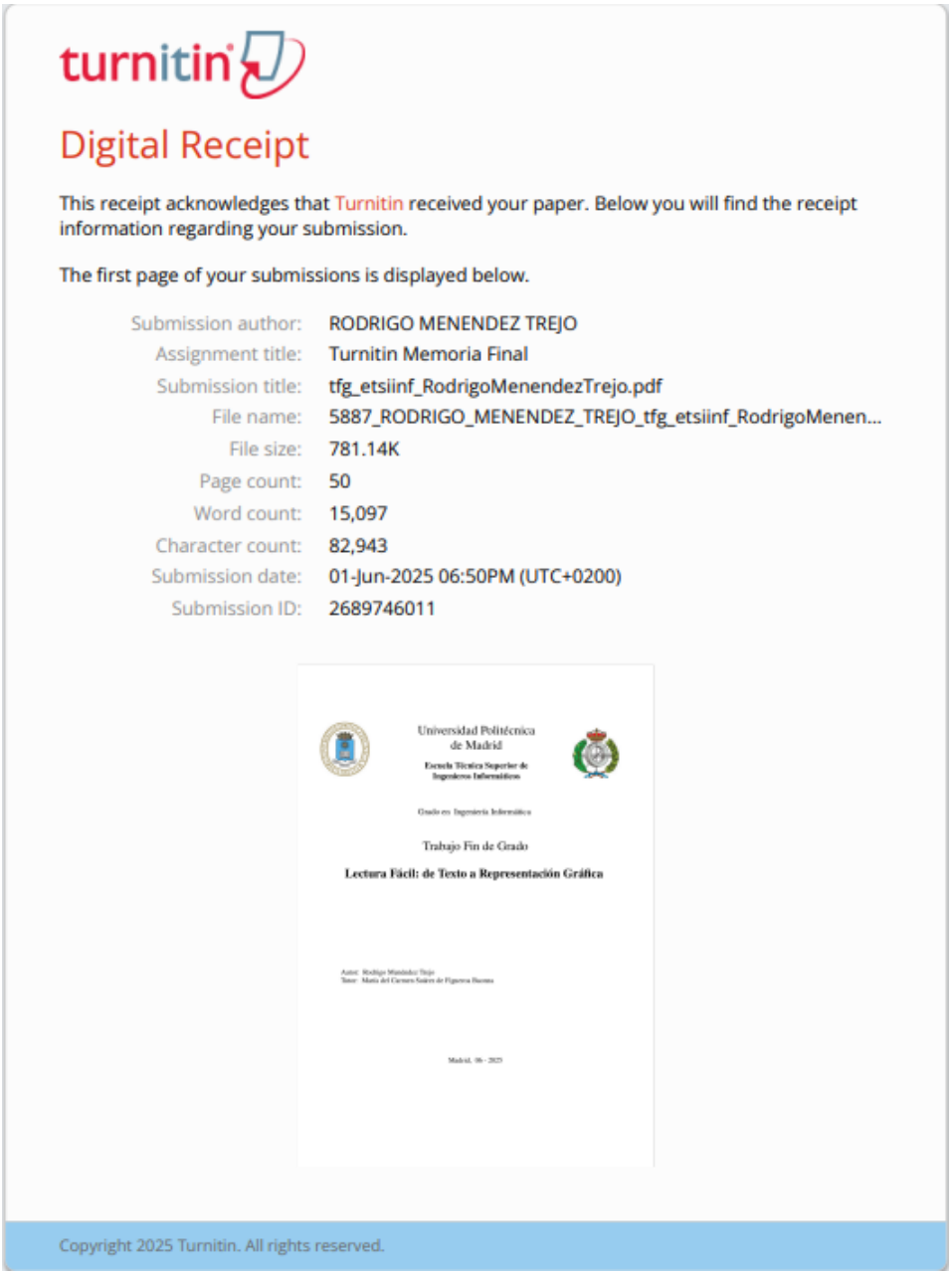
[23] Varun Shenoy. *GraphGPT: Extrapolating Knowledge Graphs from Unstructured Text using GPT-3*. [En línea]. Disponible: https://github.com/varunshenoy/GraphGPT. Accedido en Febrero de 2025. 2023.

[24] OpenAI. *Precios*. [En línea]. Disponible: https://openai.com/es-ES/api/pricing/. Accedido en abril de 2025.

[25] Greggory Elias. *Las 10 mejores técnicas de prompting para LLM en 2025 - Skim AI*. [En línea]. Disponible: https://skimai.com/es/10-mejores-tecnicas-de-incitacion-para-llms-en-2025/. 2025.

Anexo

Anexo A. Informe de originalidad



turnitin

Digital Receipt

This receipt acknowledges that Turnitin received your paper. Below you will find the receipt information regarding your submission.

The first page of your submissions is displayed below.

Submission author:	RODRIGO MENENDEZ TREJO
Assignment title:	Turnitin Memoria Final
Submission title:	tfg_etsiinf_RodrigoMenendezTrejo.pdf
File name:	5887_RODRIGO_MENENDEZ_TREJO_tfg_etsiinf_RodrigoMenen...
File size:	781.14K
Page count:	50
Word count:	15,097
Character count:	82,943
Submission date:	01-Jun-2025 06:50PM (UTC+0200)
Submission ID:	2689746011

Top Sources

The sources with the highest number of matches within the submission. Overlapping sources will not be displayed.

1	Student papers	
Universidad Politécnica de Madrid		1%
2	Student papers	
Universidad Internacional Isabel I de Castilla		<1%
3	Student papers	
Universidad de Salamanca		<1%
4	Student papers	
ucb		<1%
5	Student papers	
UNIBA		<1%
6	Student papers	
Universidad de los Hemisferios		<1%
7	Student papers	
Systems Link		<1%
8	Student papers	
Escuela Politecnica Nacional		<1%
9	Student papers	
Universidad Carlos III de Madrid		<1%
10	Student papers	
Universidad Católica San Pablo		<1%
11	Student papers	
University of Technology, Sydney		<1%

2% Overall Similarity

The combined total of all matches, including overlapping sources, for each database.

Filtered from the Report

- Bibliography
- Quoted Text

Top Sources

0% Internet sources
0% Publications
2% Submitted works (Student Papers)

Este documento esta firmado por

	**Firmante**	CN=tfgm.fi.upm.es, OU=CCFI, O=ETS Ingenieros Informaticos - UPM, C=ES
	Fecha/Hora	Tue Jun 03 18:14:10 CEST 2025
	Emisor del Certificado	EMAILADDRESS=camanager@etsiinf.upm.es, CN=CA ETS Ingenieros Informaticos, O=ETS Ingenieros Informaticos - UPM, C=ES
	Numero de Serie	561
	Metodo	urn:adobe.com:Adobe.PPKLite:adbe.pkcs7.sha1 (Adobe Signature)