Escuela Superior
de Ingeniería y Tecnología
Universidad de La Laguna

Trabajo de Fin de Grado

Grado en Ingeniería Informática
Itinerario de Tecnologías de la Información

Procesamiento de Lenguaje Natural sobre textos antiguos

Natural language processing over ancient texts

Sergio Delgado López
(alu0100893601@ull.edu.es)

San Cristóbal de La Laguna, a miércoles 8 de agosto de 2021

Dña. **Isabel Sánchez Berriel**, con N.I.F. 42885838-S profesora contratada doctora adscrita al Departamento de Ingeniería Informática de Sistemas de la Universidad de La Laguna, como tutora.

C E R T I F I C A

Que la presente memoria titulada:

"Procesamiento de Lenguaje Natural sobre textos antiguos".

Ha sido realizada bajo su dirección por D. **Sergio Delgado López** (alu0100893601@ull.edu.es), con N.I.F. 78649667-V.

Y para que así conste, en cumplimiento de la legislación vigente y a los efectos oportunos firma la presente en La Laguna a 8 de agosto de 2021.

Agradecimientos

Antes de comenzar con toda la explicación y desarrollo que va a tener este documento sobre mi trabajo de fin de grado, me gustaría agradecer principalmente a mi tutora académica, Isabel Sánchez Berriel por jugar el papel de guía en este proyecto y por ser un claro ejemplo de buen docente.

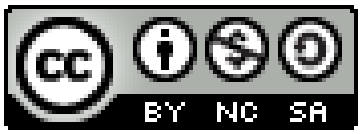
También agradecer a mi familia que me apoyara durante todos los momentos difíciles que sufrí durante el grado, animándome a no rendirme y haciéndome ver que soy capaz de hacer todo lo que me propusiera. De igual manera, a mi pareja, ayudándome a mantener los pies en el suelo cuando más lo necesitaba.

Por otro lado, a mis amigos, tantos aquellos que he conocido dentro de la universidad como los que no, algunos sirviendo como faro y otros como desahogo, pero lo más importante, pudiendo contar con ellos para lo que hiciera falta.

Por ello, si este trabajo se ha completado es gracias a estas personas.

Muchas gracias de corazón.
Sergio.

Licencias



Resumen

El presente trabajo se ha orientado a desarrollar una herramienta que mediante el uso de procesamiento de lenguaje natural, analice y textos del siglo XV y XVI en formato PDF. De forma automática se obtienen las entidades nombradas y relaciones entre ellas para almacenarlas en una base de datos orientada a grafos que facilite el posterior análisis de relaciones entre propietarios y lugares en la isla de Tenerife en la época de estudio.

Para la implementación del proyecto se ha optado por tecnologías tales como el lenguaje de programación Python dentro del entorno de programación colaborativo de Google Colab (cuadernos Júpiter), librerías de Python para el procesamiento de lenguaje como NLTK, y más herramientas que se han usado para conseguir la meta del trabajo, las cuales se verán en profundidad a lo largo de este documento.

Palabras clave*:* procesamiento de lenguaje natural, Python, NLTK, Hunspell, spaCy, Google Colab, *machine learning*, bases de datos, grafos, Neo4J.

Abstract

The present work has been oriented to develop a tool that by means of the use of natural language processing, analyzes and texts of the XV and XVI centuries in PDF format. The named entities and relationships between them are automatically obtained to store them in a graph-oriented database that facilitates the subsequent analysis of relationships between owners and places on the island of Tenerife at the time of study.

For the implementation of the project, we have opted for technologies such as the Python programming language within the collaborative programming environment of Google Colab (Jupiter notebooks), Python libraries for language processing as NLTK, and more tools that have been used to achieve the goal of the work, which will be seen in depth throughout this document.

Keywords: natural language processing, Python, NLTK, Hunspell, spaCy, Google Colab, machine learning, databases, graphs, Neo4J.

Tabla de contenido

Índice de figuras:

Índice de tablas:

Capítulo 1. Introducción.

1.1. Prefacio.

A lo largo de este documento se va a describir dentro de lo posible todo lo acontecido durante el desarrollo del trabajo de fin de grado, pasando por los elementos previos al propio trabajo como los objetivos, el análisis del problema o el estado del arte. Luego se pasará a la parte propia del desarrollo e implementación del proyecto, enumerando y describiendo los procesos llevados a cabos, y si es necesario se incluirán figuras o fragmentos de código con el fin de complementar lo que se especifique. Para concluir se tendrá un capítulo donde se expondrá si se han cumplido los objetivos y metas iniciales, también las dificultades encontradas y el futuro del propio proyecto si tuviera lugar.

1.2. Motivo del proyecto.

Este proyecto se enmarca en el contexto de los estudios sobre la historia de las Islas Canarias, sobre todo para la época de la conquista de los Reyes Católicos de España que fue un hecho importante para el devenir del territorio canario. Poder crear un sistema que recoja estos datos, y que se puedan consultar y gestionar con medios informáticos modernos implica una mejora en aspectos como disponibilidad, accesibilidad o gestión. De esto se benefician los usuarios interesados en el estudio de la historia de Canarias, y por otro lado conlleva a que esta fuente de información no se pierda con el paso del tiempo por encontrarse en medio antiguos y volátiles.

También se nutre del concepto de Humanidades Digitales, que por definición es un área que aplica los conocimientos de las nuevas tecnologías a los problemas de las ciencias humanas, en este caso la historia de Canarias. Y tiene como objetivo fundamental la creación de bases de datos con recursos digitales y generar investigación y conocimientos.

Culminar este proyecto con éxito supondría una aportación a la difusión de información histórica aprovechando el avance de las nuevas tecnologías para facilitar la gestión de la información en fuentes documentales y su exposición online, lo cual es el objetivo de este proyecto

1.3. Objetivo.

El objetivo de este proyecto es realizar un procesamiento de lenguaje natural sobre unos textos obtenidos de una biblioteca virtual y de ellos obtener datos significativos donde se distingan entidades y relaciones, y de esta manera poder generar una base de datos orientada a grafos donde se puedan diferenciar nodos y arcos que describan lo que reside en el texto.

Los textos son transcripciones literales de las datas de Tenerife de los siglos XV y XVI tales como herencias, ventas u otros datos que se recogían antiguamente en los libretos de los notarios y escribanos de las islas canarias en estos siglos.

Para llevar a cabo todo el proceso de análisis del lenguaje, se ha optado por trabajar con Python, ya que tiene un amplio repertorio de librerías dentro de este campo. Destacar también que para el uso de una base de datos orientada a grafos se ha preferido trabajar con algo de mayor nivel y usar herramientas ya creadas que proporcionen lo que se busca para este trabajo.

1.4. Alcance.

Se pretende desarrollar un sistema que permita extraer de un texto los datos que se consideren relevantes y con ellos crear una base de datos de la cual se pueda obtener un beneficio real. Principalmente se ha trabajado con una muestra singular del texto, pero en el caso de que el proyecto sea exitoso, el modelo se mantiene y funcionaría con otros tipos de textos distintos, de los que obtener datos diferentes, pero también importantes, todo dependiendo del uso que se quiera hacer del sistema creado.

Así mismo, se pueden enumerar una serie de hitos que sirvan de referencia para determinar que el proyecto se ha realizado exitosamente, entre estos se pueden enumerar:

- ➢ Crear un sistema que consiga extraer los datos de un texto que se le introduce en formato PDF (tiene que ser formato PDF real, no valen imágenes). No se plantean restricciones respecto al contenido, siempre y cuando el texto esté dentro de un marco para el cual ha sido pensado el sistema implementado.
- ➢ Facilitar la posibilidad de obtener distintos tipos de datos del texto a analizar, ya que no siempre tiene que interesar crear una base de datos, por ello se obtendrán del texto diferentes formas de división del mismo, o distintos resultados dependiendo de la herramienta especificada.

- ➢ Crear una base de datos orientada a grafos donde se presenten los datos de manera visual que facilite el análisis a los usuarios.
- ➢ Uso de un entorno colaborativo donde el código y las implementaciones del mismo se puedan modificar al momento, y también que cualquier usuario pueda modificarlo si lo desea.

Más adelante se nombrarán y describirán todas las herramientas, tecnologías y librerías usadas para este trabajo.

1.5. Estado del arte.

Atendiendo simplemente a la parte de Procesamiento de Lenguaje Natural (PLN), hay innumerables proyectos, más o menos famosos, que tratan sobre analizar el texto y extraer de esta información relevante. De igual manera esto no solo se aplica a textos físicos, sino también al reconocimiento de voz que al mismo tiempo repercute en el campo de la inteligencia artificial. Por ello se puede decir que, si solo se hace hincapié en el PLN, el estado del arte es cuantioso y bien conocido por la comunidad.

En cambio, si se profundiza en el objeto de estudio que tiene este proyecto, este aspecto cambia, puesto que es algo con matices que invitan a la peculiaridad del trabajo, es decir, se hace un procesamiento de lenguaje sobre textos transcritos antiguos con lo que eso conlleva (fallos de ortografía, de puntuación, inclusión de caracteres especiales, etc.). Sin embargo, el principal hándicap es que están escritos en lenguaje castellano antiguo, por lo que la existencia de herramientas que contemplen este escenario es reducida. Básicamente es ahí donde reside la dificultad y el reto de este proyecto.

Existen algunos proyectos que plantean una escena similar y que son relativamente actuales. El primero de ellos se trata de BiographyNet [1], que es un proyecto que trata de extraer relaciones entre personas, lugares y eventos, entendiendo esto como historia electrónica multidisciplinar. Este tiene como metas proveer un mayor conocimiento histórico desde un portal biográfico, también inspirar a nuevos historiadores a investigar en nuevos proyectos, y por último ofrecer recursos reutilizables para la comunidad. Dentro de BiographyNet se usan herramientas como OpeNER, HeidelTime, Ontotagger o sistema de UKB Word Sense Disambiguation. El segundo proyecto es el Dutch Biography Portal [2], que se trata de una colección online de obras de referencia y conjuntos de datos actualmente dispersos en Internet, que contienen información biográfica sobre personas notables en la historia holandesa, desde los primeros tiempos

hasta el presente. Contiene colecciones impresas y online, y conjuntos de datos con información biográfica sobre los habitantes.

Por otra parte, encontramos el proyecto: Memorie di Guerra [3], desarrollado en colaboración por diferentes grupos de investigación de universidades italianas. En el proyecto se lleva a cabo un análisis computacional de textos italianos de las Primera y Segunda Guerras Mundiales, en el que se aplican técnicas del procesamiento del lenguaje natural para obtener información estructurada a partir de ellos.

De resto, no se han encontrado proyectos similares a este, o por lo menos no existen de manera online.

1.6. Destinatarios.

Los principales beneficiados del proyecto son dos, los primeros son aquellas personas interesadas en conocer los datos que se han estudiado sobre los textos. Estos pueden ser historiadores o entusiastas del tema que quieran beneficiarse del sistema para llevar a cabo sus propios estudios. El segundo grupo es la comunidad de usuarios que tienen inclinaciones hacia el estudio del procesamiento del lenguaje natural, siendo este proyecto una muestra del uso de las herramientas disponibles y de los que se puede conseguir haciendo uso de estas técnicas.

Por otro lado, este proyecto puede satisfacer necesidades de entidades como colegios, museos o casas culturales, ya que el producto final del proyecto puede aportar una instantánea de cómo era un tiempo pasado para la situación socioeconómica.

La herramienta puede interesar en aquellos casos en los que se quiera:

A. Consultar información recogida en escritos del pasado de las Islas Canarias.
B. Compartir cómo se ha llevado a cabo el análisis del lenguaje y como se han usado las herramientas.
C. Establecer pautas o conductas del pasado.
D. Elaboración de un sistema atractivo para mostrar información del pasado.

Capítulo 2. Análisis del problema.

En este apartado se va a describir el planteamiento inicial del trabajo, en este caso es la definición del problema que se quiere solucionar con este proyecto, y también el análisis de los textos con los que se va a trabajar.

2.1. Fuentes documentales.

Los textos objetos de estudio se han obtenido de la biblioteca virtual de Viera y Clavijo [4], que pertenece al Instituto de Estudios Canarios. Los textos forman parte del proyecto *Fontes Rerum Canarium*, que es la mayor colección de textos y documentos de la historia de Canarias que existe en las islas. Entre los textos recogidos se cuentan títulos en los que han trabajado especialistas de reconocido prestigio, se incluyen las primeras crónicas de la conquista o las ediciones de crónicas canarias inéditas. La colección en formato digital permite un acceso directo a lo que fueron los primeros años del Archipiélago, con el interés que poseen estos textos desde el punto de vista histórico, artístico, lingüístico o etnográfico.

Los dos textos que se analizaron inicialmente para su tratamiento fueron los siguientes:

1. Las datas de Tenerife (1978) [5]: libros I a IV de datas originales: contienen los albalaes otorgados por don Alonso Fernández de Lugo por los que, en virtud de poder especial de los Reyes, hizo repartimiento de la isla recién conquistada.
2. Protocolos de Juan Márquez (1518-1521) (Vol. I) [6]: contiene los Protocolos Notariales de la isla de Tenerife en el siglo XVI y pertenecen al escribano público de San Cristóbal de La Laguna, Juan Márquez, elegido por el Consejo de Tenerife,

2.2. Definición del problema.

Como ya se ha dejado entrever en los apartados anteriores, el problema en el que se basa este proyecto es crear una base de datos orientada a grafos a partir de la información extraída de textos escritos en castellano del s. XVI que han sido digitalizados en formato PDF. Ahora bien, para poder obtener el producto deseado, es necesario tener en cuenta una serie de aspectos.

En primer lugar, los textos no están perfectamente estructurados y contienen muchas discrepancias respecto al español actual, por ejemplo:

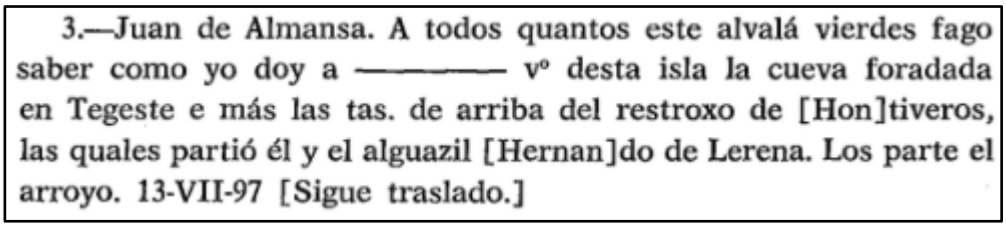

3.—Juan de Almansa. A todos quantos este alvalá vierdes fago saber como yo doy a ———— vº desta isla la cueva foradada en Tegeste e más las tas. de arriba del restroxo de [Hon]tiveros, las quales partió él y el alguazil [Hernan]do de Lerena. Los parte el arroyo. 13-VII-97 [Sigue traslado.]

Figura 1. Fragmento de texto ejemplo de los textos a analizar

Como puede observarse en la Figura 1 del texto, la anacronía del lenguaje supone un inconveniente. Si se tratase el texto en su forma original, esto daría lugar a numerosos problemas por el hecho de ser una transcripción en una versión antigua de nuestra lengua, por ello habrá que depurar el texto lo máximo posible antes de someterlo a su procesamiento automático.

Tal y como puede verse se usan términos como *quales*, *restroxo* o *vierdes*. El problema de esto que actualmente no hay herramientas que contemplen este vocabulario, por ello será necesario determinar la forma de controlar mejor esta dificultad y arrojar un mejor resultado.

En tercer lugar y probablemente el proceso más importante de todo el proyecto, es la propia extracción de los datos del texto. Tras sufrir toda la depuración, se buscará la manera de obtener los datos en el formato correcto para luego insertarlos en una base de datos. Por lo general las herramientas de procesamiento de lenguaje usan etiquetas para determinar qué es cada palabra, esto se comentará en mayor profundidad más adelante.

En cuarto lugar, se tiene que trabajar con los datos obtenidos y distinguir entre estos los que aportan valor para una base de datos, estos pueden ser los nombres propios de personas o lugares y las relaciones existentes entre ellos.

Por último, pero no menos importante, hay que tener en cuenta que, en este tipo de proyectos, la escalabilidad es muy importante, y no es lo mismo trabajar con texto de unas decenas de páginas, a hacerlo con textos de miles de páginas, por lo tanto, es necesario prestar atención a la optimización y coste computacional de la aplicación que se implemente en este trabajo.

2.3. Análisis de los textos tratados.

En un principio se han proporcionado dos tipos de textos distintos sobre los cuales trabajar, por ello antes de comenzar a desarrollar, es necesario realizar un análisis de

manera personal a los textos, para ver cuál interesa más analizar o cuál es más adecuado para trabajar.

Texto 1. *Datas de Tenerife (Libros 1ro al 4to de datas originales)*:

Este documento recoge la mayoría de los albalaes (carta real concedida por alguna merced o miembro de la realeza) otorgados por don Alonso Fernández de Lugo por los que, en virtud de poder especial de los reyes de España, hizo repartimiento de la tierra de la isla de Tenerife recién conquistada.

Extracto: "*tras la numeración, el nombre del donatario, con cualquier identificación complementaria que expresa el albalá: conquistador, oficio, naturaleza (portugués, canario, guanche, gomero); la tierra u otro bien donado (cueva, colmenas, aguas), con su localización y linderos, que, si a veces se limitan a los nombres de los dueños, a menudo se precisan con detalles topográficos; las enmiendas a rebajar del gobernador-repartidor. El nombre del destinatario, si se repite en el texto y aun el de los lindantes y otros en el mismo caso, se reducen en las menciones posteriores a una raya o a las siglas iniciales."* (Datas de Tenerife, Libros 1ro al 4to de datas originales [5])

Información de valor: El documento a trabajar tiene un total de 424 páginas, entre las cuales, la información con valor es la de los cuatro libros que poseen la información del repartimiento, en concreto la información contenida entre la página 19 (inicio del libro 1) y la 375 (fin del libro 4).

Aspectos a tener en cuenta en este documento:

1. La información útil está semiestructurada y bien identificada en el documento.
2. Dicha información está escrita en extractos, es decir, se podría dividir todo el texto, obteniendo cláusula a cláusula para su posterior procesado.
3. El uso de abreviaturas se puede tratar automáticamente para evitarlas y así paliar posibles conflictos con un analizador de texto.
4. El uso de un lenguaje antiguo: es posible que, al procesar el texto con alguna herramienta, muchas palabras no estén contempladas, pudiendo llegar a ser un problema.
5. Volumen de información a trabajar: 356 páginas de PDF.
6. Aprovechando el uso de la numeración a principio de cada extracto, se sabe dónde se encuentra la información a procesar.

Texto 2. *Protocolos de Juan Márquez, tomo I*:

Se trata de un compendio de los Protocolos Notariales de la isla de Tenerife en el siglo XVI. Se considera que se trata de una fuente insustituible para un estudio serio de la realidad cotidiana de la época. Hay recogidos 6 protocolos escritos por el escribano de La Laguna Juan Márquez, y abarcan desde septiembre de 1518 a 1521 inclusive.

La documentación extraída es bastante variada, aunque con un predominio de las obligaciones. Se ha calculado el porcentaje de cada tipo de escritura con respecto al total y vemos que las obligaciones forman el 66%, los poderes el 19% y las ventas el 3'5%.

Información de valor: El documento a trabajar tiene un total de 536 páginas, entre las cuales, la información valiosa es la que está entre las páginas 77 y 533. Esto abarca todos los protocolos escritos entre los años 1518 y 1521.

Aspectos a tener en cuenta en este documento:

1. Al igual que en el caso del documento de datas, también en este se usan abreviaturas.
2. La información se presenta en cláusulas indivisibles.
3. Hay mucha variedad de información en el texto, por lo que es altamente probable que, a la hora de procesar en busca de información valiosa para crear una base de datos orientada a grafos, gran parte sea desechada, incrementando un coste en el proceso de análisis.
4. Aprovechando el uso de la numeración a principio de cada extracto, se sabe dónde se encuentra la información a procesar.
5. Volumen de información: 456 páginas de PDF.

Conclusión:

Teniendo en cuenta que no se ha procesado ningún documento en el momento de realizar este análisis, la decisión tomada en aquí puede variar en el futuro.

Después de haber analizado los datos disponibles en cada texto, se cree que lo mejor es empezar a trabajar con el documento de *Datas de Tenerife*, ya que proporciona información que puede favorecer a la extracción de entidades nombradas y relaciones que se almacenen en una base de datos. Esta decisión se justifica en que, respecto a los demás textos, en este el problema están más controlados y mejor definidos, favoreciendo a

identificar las dificultades y las herramientas con las que afrontarlas. Textos más complejos y ambiciosos quedan fuera del alcance de este proyecto. Destacar también que el documento elegido tiene menos volumen de información (repercute en el coste computacional) y está dividido claramente en extractos.

Capítulo 3. Estudio previo.

Probablemente constituye la parte más extensa del proyecto junto al propio desarrollo del código. El estudio que se ha realizado de manera previa a todos los aspectos que engloban el trabajo como el procesamiento de lenguaje natural, el entorno de desarrollo, tecnologías y herramientas disponibles, diferentes enfoques de desarrollo, pasos a tomar, etc., han ocupado una gran parte del tiempo.

Para poder hacer frente a los objetivos propuestos, se ha llevado a cabo un análisis bastante extenso acerca de muchos puntos, tanto en el ámbito del lenguaje de programación como en el ámbito del tratamiento del lenguaje natural. Entre otras cuestiones: como se expondrá más adelante: conceptos, formas de trabajo, posibles herramientas y módulos a utilizar, capacidades y filtros de las distintas librerías y salidas correspondientes, etc.

3.1. Procesamiento de lenguaje natural.

El campo en el que radica este trabajo es el procesamiento del lenguaje natural [7], que por definición es un campo de la informática que estudia la interacción entre las computadoras y el lenguaje humano. Entre sus aplicaciones destacan el análisis del lenguaje, comprensión del lenguaje, generación de lenguajes o síntesis de voz. Por lo que es lógico que se recurra a este concepto en el trabajo, ya que se busca analizar el lenguaje de unos documentos.

Dentro del PLN se tienen los *pipelines* (serie de funciones que se aplican a un texto para obtener atributos como por ejemplo *part-of-speech* o *named-entity-recognition*). Normalmente el *pipeline* típico pasa por las fases de **tokenizer**, que trata de dividir el texto en unidades más pequeñas, **tagger** [8], donde se determina la categoría gramatical, el **parser**, encargado principalmente de determinar las frases, y luego **ner** [9], que identifica si los tokens forman parte de entidades nombradas (por lo general personas, organizaciones, lugares, etc.). Siguiendo este proceso correctamente, quedan identificadas todas las partes de un texto, para que acto seguido se realicen las operaciones pertinentes.

Por otro lado, el PLN se compone de varios procesos [10] que aportan información como resultado dependiendo del tipo de análisis que se quiera realizar. Estos son:

- **Análisis morfológico**. El análisis de las palabras para extraer raíces, rasgos flexivos, unidades léxicas compuestas y otros fenómenos.
- **Análisis sintáctico**. El análisis de la estructura sintáctica de la frase mediante una gramática de la lengua en cuestión.
- **Análisis semántico**. La extracción del significado de la frase, y la resolución de ambigüedades léxicas y estructurales.
- **Análisis pragmático**. El análisis del texto más allá de los límites de la frase, por ejemplo, para determinar los antecedentes referenciales de los pronombres.
- **Planificación de la frase**. Estructurar cada frase del texto con el fin de expresar el significado adecuado.
- **Generación de la frase**. La generación de la cadena lineal de palabras a partir de la estructura general de la frase, con sus correspondientes flexiones, concordancias y restantes fenómenos sintácticos y morfológicos.

3.1.1. Técnicas de PLN.

En relación con lo descrito en el apartado previo, en este proyecto en un principio se buscará usar el análisis morfológico en el apartado del depurado y preprocesamiento de los textos, luego el análisis sintáctico para determinar las estructuras sintácticas, y por último el análisis semántico para comprender el significado de la frase, aprovechando así para extraer las posibles relaciones y datos valiosos de los textos.

3.2. Tecnologías y entorno de trabajo.

En este apartado se van a nombrar y describir dentro de lo posible las tecnologías adoptadas para el desarrollo del sistema de procesamiento de texto, abarcando tanto los aspectos del código, como bases de datos o el propio entorno de trabajo.

3.2.1. Google Colaboratory.

En lugar de trabajar en un entorno local, como por ejemplo con un IDE o con Visual Studio Code, se ha optado por trabajar dentro del entorno de *cloud computing* (servicios en la nube) Google Colaboratory [11], que proporciona los recursos necesarios para que la ejecución de la aplicación no se vea limitada por las especificaciones de la máquina. También aporta un entorno dinámico y colaborativo de desarrollo, pudiendo vincular este servicio con el sistema de archivos de Google Drive. Por lo tanto, todo el proyecto reside en la nube de Google, esto significa que puede modificarse y ejecutar desde cualquier lugar con conexión a internet.

Destacar también que en este entorno colaborativo se puede incorporar texto en el que poder separar el contenido del código fuente y explicar lo que se realiza en cada sección.

En el Capítulo 10. Anexo. de este proyecto se puede encontrar un enlace al proyecto dentro de Google Colab.

3.2.2. Python.

De entre todos los lenguajes de programación que existen para realizar este tipo de aplicaciones, se ha escogido Python [12] tanto por el abanico de posibilidades que ofrece, como por ser reto a nivel personal, puesto que es un lenguaje cuyo uso se ha universalizado y no se imparte en las asignaturas cursadas en el Grado. Decir también que, Python y Google Colaboratory conviven muy bien, siendo una solución óptima para este proyecto.

A modo de definición, Python constituye un lenguaje de programación de alto nivel, interpretado y multipropósito. Su filosofía hace hincapié en una sintaxis que favorezca un código legible, siendo esta clara y concisa, y cuya potencia y flexibilidad lo convierten en un lenguaje muy productivo. Soporta orientación a objetos, programación imperativa y funcional, es multiplataforma y utiliza un tipado dinámico (comúnmente denominado lenguaje débilmente tipado).

De igual manera, Python tiene una gran comunidad, tanto para el campo de procesamiento de lenguaje natural, como para cualquier aspecto propio del lenguaje que tenga que ver con problemas y soluciones de algoritmos.

Por último, destacar que, la gran mayoría de herramientas o aplicaciones grandes del mercado, tiene su versión o adaptación para usarse en Python, lo cual es una ventaja significativa.

3.2.3. Herramientas de procesamiento de lenguaje natural.

Herramienta	Descripción	Uso
PyPDF2 [13]	Librería de Python que tiene un conjunto de herramientas capaces de trabajar con PDF	Transformar PDF a TXT
NLTK [14]	NLTK es una plataforma líder para crear programas Python que funcionen con datos de lenguaje humano.	*Tokenize* y *stopwords*,
Hunspell [15]	Librería de corrección ortográfica.	Revisión ortográfica
spaCy [16]	spaCy es una biblioteca para el procesamiento avanzado del lenguaje natural en Python y Cython. Se basa en las últimas investigaciones y se diseñó desde el primer día para utilizarse en productos reales.	*Tokenize, lemmatization, part-of-speech* y otros

Tabla 1. Herramientas de procesamiento de lenguaje.

En esta Tabla 1 solo se han presentado las herramientas más significativas usadas en la aplicación, en los apartados siguientes se detallará más el uso de estas y los algoritmos que se han implementado para obtener los resultados.

3.2.4. Base de datos orientada a grafo.

Las Bases de Datos Orientadas a Grafos [17] representan la información como nodos de un grafo y las relaciones son sus aristas, de esta manera se puede aplicar la teoría de grafos a la base de datos. Este tipo de BBDD proporciona la capacidad de detectar conexiones o analizar los datos en función de las relaciones que hay entre ellos. También debido a que se almacenan las relaciones explícitamente, los algoritmos se ejecutan en un tiempo inferior. La naturaleza de esta clase de BBDD es perfecta para el propósito que se busca en este proyecto, donde el objeto principal es el estudio de las relaciones entre entidades nombradas que puedan extraerse del texto.

Dado que trabajar con la base de datos no es el objetivo principal de este proyecto, se ha buscado una solución sencilla y que cumpla con lo necesario para almacenar y mostrar los datos con los que se trabaja. En este caso se ha recurrido a la herramienta Neo4J [18], que es una plataforma que brinda a los desarrolladores las herramientas más fiables y potentes para fácilmente construir aplicaciones inteligentes y machine learning.

Lo más importante que aporta a este trabajo Neo4J es que permite almacenar de forma sencilla y visual la relaciones entre personas y lugares que se extraen de los textos. Otra

razón por lo que se ha adoptado en este proyecto, es porque tiene integración sencilla con Python y también posee un servicio gratuito en la nube donde poder trabajar con la base de datos de manera remota y visualizar el grafo de manera inmediata, repercutiendo lo mínimo posible en la aplicación en sí.

Esta herramienta online dentro de Neo4J es Neo4J Aura, que proporciona al usuario la capacidad de trabajar en una plataforma online con sus bases de datos en lugar de tener que montarlas en máquinas locales. No es necesario tener que preocuparse por la administración de las BBDD y dispone de seguridad de alto nivel, también funciona con Cypher, que es uno de los lenguajes de consulta de grafos más usados.

Capítulo 4. Diseño e implementación del procesamiento y análisis del texto.

Este es el capítulo más extenso de todo el documento, puesto que es donde se van a describir y detallar dentro de lo posible los procesos llevados a cabo, con qué herramientas y con qué algoritmos, para ello se va a dividir la explicación según los aspectos más significativos, y dentro de estos profundizar en lo implementado. Pero antes, se enumeran los pasos seguidos a lo largo del desarrollo de este proyecto.

4.1. Pasos a seguir.

I. Análisis previo de los textos para decidir con cuál trabajar.
II. Configuración de Google Colab y Python.
III. Sincronización del almacenamiento de Google Drive con Google Colab.
IV. Transformación del texto PDF en formato TXT.
V. Depurar y preprocesar el texto a fin de que quede limpio y favorezca el procesado automático del lenguaje mediante herramientas.
VI. Analizar el texto depurando las formas canónicas.
VII. Analizar el texto en busca de entidades nombradas (etiquetado de las partes del texto mediante procesamiento de lenguaje).
VIII. Extracción de los datos relevantes para la creación de una base de datos.
IX. Verificación y corrección de los datos extraídos.
X. Establecer comunicación entre la aplicación Python y la base de datos Neo4J.
XI. Elaboración de la memoria, informes y otros documentos pertinentes (aunque esta tarea se realiza de manera continua mientras se desarrolla el proyecto).
XII. Presentación de los resultados.
XIII. Corrección de errores y refactorización del código (si fuera necesario).

4.2. Depuración previa del texto.

Tal y como se ha descrito anteriormente, el objeto sobre el que se trabaja en este proyecto son textos que probablemente contengan errores de impresión o escritura, eso podría implicar que las herramientas de procesamiento usadas desemboquen en errores. Por ello conforme se iba analizando el texto se determinaron los aspectos a corregir para paliar equivocaciones.

Con la variedad de herramientas usadas, lo mejor en este caso es describir cada proceso y algoritmo por separado, aun cuando el resultado de unos depende directamente de otros,

por el hecho de que el texto a trabajar es el mismo que se va transformando por el *pipeline* por el que se ha optado. De esta manera, se van a enumerar según las funciones del código.

A. Texto original en formato PDF:

Problema: El texto nativo se encuentra en formato PDF, y las herramientas de procesamiento y análisis no conviven bien con dicho formato.

Para solucionar esto es necesario transformar el texto a formato TXT, es decir texto plano, que es fácilmente entendible por todas las herramientas que se usan en el código. Para poder obtener este resultado se ha recurrido a la librería de Python PyPDF2, que permite convertir texto en PDF a formato TXT de manera bastante sencilla. En este caso es necesario que el texto del PDF sea texto propiamente, y no imágenes de texto.

Dentro de esta librería hay muchas funciones interesantes, pero en este caso sencillamente se le pasa el texto que se desea transformar y se le indica desde qué página se empieza a leer y cuántas páginas. Para esto se ha recurrido a `PyPDF2.PdfFileReader(pdf_file_object)` donde se indica el fichero de lectura, y luego `pdf_reader.getPage(pag)` que es la función encargada de leer las páginas. El resultado de esta función es una variable cadena donde reside todo el texto leído, que es el que se usará a lo largo de toda la implementación.

B. Presencia de abreviaciones en el texto:

Problema: Dentro de los documentos se indica que hay abreviaciones y de igual manera el significado de estas. Como estas abreviaciones son particulares del texto, es necesario sustituirlas por su significado original.

A fin de solucionar esto no ha sido necesario recurrir a ninguna herramienta que no fuera propiamente Python. Cuando en el texto se encuentra una abreviación, esta tiene que ser sustituida por su valor original.

C. Depuración de *stopwords*:

Problema: Entiéndase por *stopwords* [19] aquellas palabras contenidas en el texto que no aportan semántica a la hora de analizarse. En general suelen ser artículos, conectores como preposiciones, conjunciones, etc. Por ello no interesa tener estas palabras dentro del documento, ya que cuando escala a un mayor tamaño, el volumen que ocupan estas

palabras puede llegar a ser considerable y son palabras funcionales más que de contenido semántico.

Muchas herramientas contienen un listado de *stopwords* listos para usarse, en este caso la librería NLTK aporta una lista para el castellano, pero aun así se ha decidido completarla combinándola con otras listas, para asegurar un barrido de palabras mayor para todo el texto.

Atendiendo a la implementación de esto, usando la función de NLTK que divide el texto por palabras y gracias a un bucle se comprueba que las palabras no estén dentro del conjunto de *stopwords*. También ha sido necesario particularizar los casos en los que a las palabras les sigue un signo de puntuación. En este caso, se puede llegar a interpretar como una nueva palabra, para no perder el formato del texto se ha asegurado de mantener la puntuación (igualmente con los espacios en blanco). En el caso de que el resultado contenga alguna palabra que se considera vacía, se podría refinar más el listado e incluir más entradas.

D. Depuración de caracteres alfanuméricos:

Problema: Dentro del texto pueden haberse heredado caracteres salvajes (p. ej. $, *) debido a las diversas transcripciones y diferentes autores.

Esta filtración engloba muchos caracteres, por ello es mucho más sencillo filtrar mediante los que es necesario que permanezcan para que el texto no pierda significado, más específicamente los alfanuméricos y los signos de puntuación esenciales. Como en los casos anteriores, también se ha recurrido a filtrarlos para los que se han usado `word.isalnum()` y `re.match("(\.$)|(\,$)|(\:$)", word).`

E. Depuración de signos de puntuación incorrectos:

Problema: Por culpa de todas las funciones llevadas a cabo hasta este punto, es posible que se hayan quedado signos de puntuación erróneos (p. ej. una frase con cuatro puntos seguidos, o dos comas juntas).

Este problema se ha afrontado analizando aquellos signos de puntuación que aportan valor semántico al texto, descartando aquellos que no lo hacen. Destacar que pueden existir puntos que refieren al uso de siglas, como por ejemplo señor y sr., por lo que se ha intentado distinguir esto para no perder información en el documento. Se ha usado al igual

que para el caso anterior filtrar por alfanuméricos y con expresión regular para los signos de puntuación.

F. Corrección ortográfica:

Problema: En este punto con el texto ya depurado, puede darse el caso de que la ortografía del texto sea errónea, bien por el origen del texto, o bien por todas las modificaciones que se han hecho hasta aquí.

Se ha optado por usar la herramienta Humspell de entre todas las disponibles para llevar a cabo una comprobación ortográfica del texto, porque esta tiene librería para Python y es muy fácil de usar. El comportamiento de esta herramienta es el siguiente: para cada palabra comprueba si tiene alguna sugerencia, en caso de tener una o varias, estas se almacenan dentro de un *array*, y el que ocupe la primera posición será la mejor sugerencia (siguiendo el criterio establecido por Humspell). Cuando tenemos una mejor sugerencia, se adopta y se sustituye por la palabra original.

Es posible también que las palabras leídas o escritas tengan conflicto en la codificación o decodificación respecto a Humspell, por ello se ha asegurado que todas las palabras tengan codificación "utf-8" usando las funciones Python de `decode("utf-8")` y `encode("utf-8").`

Una vez el texto ya ha sufrido todos estos procesos descritos en este apartado, se puede concluir que el documento ha sido depurado con un alto grado de éxito, minimizando al máximo posible fuentes de errores de cara a la ejecución. A continuación, es necesario describir una serie de aspectos derivados de la depuración del texto:

Concepto	Descripción
Formato del texto	Tras ejecutarse cada función, ha sido primordial fijarse en el formato en el que se encuentra el texto tratado para que de esta manera se eviten errores en pasos posteriores. Por formato correcto se refiere a la puntuación, erratas, mayúsculas o minúsculas, separación por palabras o frases, etc.
Almacenamiento de los datos	Aunque el texto se almacene localmente en el entorno, también se almacena en ficheros, puesto que muchos procesos generan el mismo resultado una y otra vez. Por ejemplo, para un texto dado, si no cambia con el tiempo, el preprocesado solo es

	necesario realizarlo una vez, teniendo almacenado los distintos resultados para las depuraciones que se hacen.
Coste computacional	Por trabajar en un entorno de *cloud computing* el coste computacional solo se ve reflejado en el tiempo que tarda en ejecutarse cada fragmento de texto, por ello, cabe decir que el tiempo de ejecución de todos los métodos de procesamiento está en 12 minutos aproximadamente, y dentro de estos, el que ocupa casi la totalidad de esta ventana temporal, es la corrección ortográfica.
Google Colab	La implementación de todos estos métodos y procesos se han llevado a cabo en Google Colab, como ya se ha comentado. Por lo que el código queda a disposición de todos los interesados en el cuaderno Júpiter de la plataforma. Cuaderno con el código disponible en el anexo.

Tabla 2. Aspectos del preprocesamiento.

4.3. Procesamiento del texto.

Una vez se ha revisado y depurado el texto, tal y como se ha explicado a lo largo del apartado de 4.2. Depuración previa del texto., es hora de pasar a un análisis más enfocado al propio procesamiento del lenguaje natural. En este problema se ha aplicado el proceso de lematización y el reconocimiento de entidades nombradas. Gracias a esto es como se obtienen los datos que se buscan para crear una base de datos orientada a grafos.

Data la importancia de esta parte del proyecto, se realizó un análisis exhaustivo de las herramientas a usar, con ensayos sobre escenarios similares, para poder determinar la que mejor se adapta a nuestro caso. Las tecnologías examinadas aportaban todas las funcionalidades necesarias para este trabajo, así que se trató de medir mediante su capacidad de enfrentarse al lenguaje castellano antiguo y los resultados arrojados.

Concepto	Descripción
Freeling	A pesar de tener una gran reputación en cuanto al análisis del español, se descartó rápidamente para este trabajo dadas las dificultades que se tuvieron para instalar la herramienta y trabajar con ella desde Python, tanto en local como en el entorno de Google Colab.

spaCy	spaCy tiene funcionalidades tanto para la lematización como para el reconocimiento de entidades nombradas, aparte de tener compatibilidad con Python y tener dependencias para el lenguaje castellano. También se trata de un proyecto firmemente reconocido, con una fuerte comunidad y completamente actual. Por otro lado, atendiendo al reconocimiento de entidades, el código de etiquetación para las palabras es bastante intuitivo.
NLTK	NLTK es una de las librerías más populares para el procesamiento de lenguaje natural, por ello se usa en casi todas las funciones implementadas en este proyecto, ya que tiene un amplio repertorio de funciones que ayudan al manejo del texto. Al igual que spaCy también tiene funciones para la lematización y para el reconocimiento, pero a la hora de tratar con el castellano no es solvente. De igual manera, las etiquetas de reconocimiento de entidades no son intuitivas.
Hunspell	El principal propósito de este es la corrección ortográfica, pero tras analizar la herramienta, se descubrió que tiene funciones de lematización para castellano, aunque este proceso no aportó un gran resultado.

Tabla 3. Análisis de tecnologías de procesamiento de lenguaje natural.

Después de haber trabajado y analizado las distintas tecnologías nombradas en la Tabla 3, la decisión de cara a la lematización y al reconocimiento de entidades ha sido usar spaCy, por las siguientes razones:

- En cuanto a tiempo de ejecución es el más rápido.
- El proceso de instalación de las dependencias y uso de funciones es más sencillo.
- Documentación completa y clara.
- Comunidad.
- La etiquetación del reconocimiento es clara e intuitiva.

Aun así, en el proyecto se usa NLTK para las funciones de manejo de texto (p. ej. *tokenize*) y Hunspell para la corrección ortográfica.

4.3.1. Proceso de *lematización*.

Se entiende por lematización [20] al proceso lingüístico que consiste en, dada una forma flexionada de una palabra (es decir, en plural, en femenino, conjugada, etc.), hallar

el lema correspondiente. El lema es la forma que por convenio se acepta como representante de todas las formas flexionadas de una misma palabra. Llevar esto a cabo favorece mucho al documento con el que se trabaja, puesto que se reduce considerablemente su volumen y se unifican las palabras en una sola que aporta la misma información semánticamente, lo que se traduce en mejor coste computacional, aparte de ser más sencillo para las herramientas de procesamiento.

Tal y como se ha detallado en el apartado previo, para el proceso de lematización se ha optado por usar spaCy, como una de sus ventajas era la sencillez, aquí queda demostrado, puesto que con la potencia de esta herramienta este proceso se realiza con unas pocas líneas de código.

Lo importante de este proceso es obtener el texto con el mismo significado y sin perder valor, pero mucho más ligero, ya que se ha dejado el lema de la palabra.

Profundizando un poco más dentro de este proceso hay dos conceptos a tener en cuenta, el primero de ellos es seleccionar el procesador de spaCy encargado de lematizar, esto se consigue con:

```
# processor selector
nlp = spacy.load("es_dep_news_trf")
lemmatizer = nlp.get_pipe("lemmatizer")
```

Figura 2. Fragmento de código lematización 1.

spaCy tiene las dependencias necesarias para trabajar con el español, tan solo basta con cargarlas y luego la propia librería se encarga de usarlas. Por otro lado, hay que indicar el lematizador para que luego lo use para cada párrafo a analizar. El objeto *nlp* es donde se instancia lo referente al texto y su procesamiento.

```
# spaCy stem process
text = nlp(no_stem_spacy_paragraphs)
result = [token.lemma_ for token in text]
```

Figura 3. Fragmento de código lematización 2.

El segundo fragmento (Figura 3. Fragmento de código lematización 2.) se puede ver como procede a almacenar todo el texto lematizado en la variable resultado, que más adelante contendrá el resultado final del texto.

Tras la ejecución del método, que se realiza sobre todos y cada uno de los párrafos extraído del texto, se obtiene el texto final igualmente divido por párrafos, que se usará

para el reconocimiento de entidades. El tiempo de ejecución de esta función en Google Colab, es de alrededor de unos 8 minutos.

4.3.2. Proceso de *reconocimiento de entidades* según su categoría gramatical.

Antes de profundizar en la implementación de cómo se ha obtenido la información, es importante saber lo que interesa de la misma, y el formato que sería adecuado para trabajar luego con la base de datos. Tal y como se ha indicado desde el principio de este documento, lo que se trata es de obtener unos determinados datos de los documentos, que según el estudio realizado son del tipo: nombres de personas, lugares u organizaciones, y relaciones, aunque bien es cierto que lo extraído de los textos no se ven en el texto original como tal, se ha optado por tomar como entidades a las personas o lugares y como relación a la acción que se llevan a cabo. Un ejemplo de esto puede ser "*Francisco López entrega a Juan Marcos*", donde las entidades son los nombres propios de la frase, y la relación sería el verbo.

Teniendo en cuenta lo anterior, se ve que es necesario reconocer las entidades según su significado y categoría gramatical, para ello es necesario identificar cada palabra. Esto se consigue gracias al análisis de la herramienta spaCy conocido como *part-of-speech*, analiza cada palabra y aporta información tal como su propio texto literal, su lema, su posición en la sintaxis de la frase y también la etiqueta de qué es la palabra, es decir, si es un verbo, un nombre propio, un artículo, etc., a continuación, se proporciona una lista de estas etiquetas:

ETIQUETA	SIGNIFICADO	ETIQUETA	SIGNIFICADO
ADJ	*adjective*	NUM	*numeral*
ADP	*adposition*	PART	*particle*
ADV	*adverb*	PRON	*pronoun*
AUX	*auxialry adverb*	PROPN	*proper noun*
CONJ	*coordinating conjunction*	PUNCT	*punctuation*
DET	*determiner*	SCONJ	*subordinating conjunction*
INTJ	*interjection*	SYM	*symbol*

NOUN	*noum*	VERB	*verb*

Tabla 4. Etiquetado del reconocimiento de entidades con spaCy.

En este caso se ha optado por almacenar toda esta información dentro de un objeto de tipo JSON [21], ya que Python convive muy bien con este formato y además no supone un coste muy grande en cuanto a almacenamiento. El objeto tendría este formato:

```
{
	"TEXT": "Almansa",
	"VALUES": {
		"LEMMA": "Almansa",
		"POS": "PROPN",
		"TAG": "PROPN"
	}
}
```

Una vez conocido los datos que interesan y el formato de estos, se puede profundizar en cómo se ha implementado. Con spaCy, el modo de operar es como se explicó en el apartado de lematización, pero en este caso no solo interesa el lema de la palabra, sino los parámetros que definen la arquitectura de la palabra, de esta manera, gracias a los tokens de spaCy, se calculan de la siguiente forma:

```
d['TEXT'] = unicodedata.normalize('NFKD', token.text).encode('ascii', 'ignore').decode('utf-8')
d['VALUES']['LEMMA'] = unicodedata.normalize('NFKD', token.lemma_).encode('ascii', 'ignore').decode('utf-8')
d['VALUES']['POS'] = token.pos_
d['VALUES']['TAG'] = token.tag_
```

Figura 4. Fragmento código reconocimiento de entidades.

Por lo tanto, simplemente hay que ejecutarlo para cada palabra y almacenar los datos en un diccionario, en este caso el JSON nombrado anteriormente. Destacar también que, para no almacenar datos en formato incorrecto, es necesario especificar la codificación del texto a la hora de introducirlo.

Tiene lugar decir que todo lo desarrollado hasta llegar al punto de almacenar los datos de reconocimiento de entidades, es decir todo el proceso de depuración del texto y la lematización, ha sido para que dicho reconocimiento se lleve a cabo con un índice de error lo más pequeño posible.

Por consiguiente, si la ejecución del reconocimiento de categorías gramaticales para las palabras se ha llevado a cabo sin fallos, se ha obtenido un diccionario donde están contenidos todos los datos necesarios para trabajar con la base de datos.

4.4. Obtención de los datos.

Hasta este punto el desarrollo del trabajo ha transcurrido dentro de lo esperado, pero es ahora cuando se determina que los datos extraídos después del procesamiento de lenguaje natural tienen el valor que se esperaba obtener. Por ello, es en este proceso donde empiezan a aparecer dificultades por las inconveniencias ya nombradas con anterioridad: las herramientas usadas no proporcionan el resultado deseado. Aunque se haya logrado establecer un diccionario con las palabras, no se han conformado como un texto donde se reconocieran las entidades nombradas (*NER - Named Entity Recognition*), quedándose en el proceso de reconocimiento de categorías gramaticales. Esto supone un problema para obtener las relaciones con las que se deseaba trabajar en una base de datos. Para subsanar esto, se tendría que haber implementado otras soluciones tecnológicas como la creación de un modelo del lenguaje, recomendada por la tutora del trabajo. Debido a las dificultades surgidas en la depuración de los textos a la hora de obtener una versión aproximada del español actual, y dadas las limitaciones de tiempo, se opta por aplicar un proceso simplificado para la obtención de entidades y relaciones entre ellas. Lo que se describe en este epígrafe es una aproximación a lo que se esperaba obtener en un principio en este trabajo.

Dada la anacronía del lenguaje de los documentos, el proceso ya descrito en el que se crea un diccionario del texto resulta no ser del todo correcto, ya que spaCy reconoce categorías gramaticales de las palabras, pero algunas son erróneas, y esto se suma al problema de no tener el proceso NER.

Por ello se ha optado por acotar en gran medida los datos intervinientes en las relaciones, buscando algo similar a lo que se muestra en la Figura 5. Ejemplo de relación.

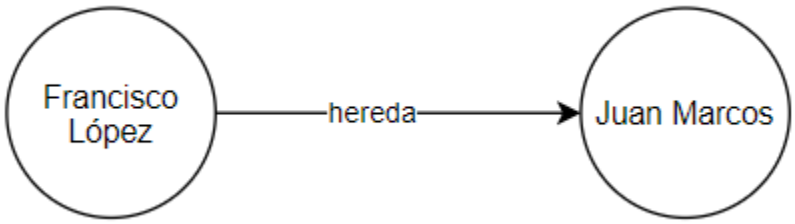


Figura 5. Ejemplo de relación.

Desde el principio del documento se ha explicado que la información se divide en extractos, luego, para obtener este resultado se ha procedido recorriendo los objetos que contienen los datos de los párrafos y analizando cada palabra para poder encontrar una coincidencia con el siguiente esquema: **A: entidad_1 → relation: relación → B: entidad_2**, restringiendo que las entidades sean nombres propios, y la relación un verbo. Dado que no se ha obtenido el proceso NER, aquí se ha elaborado un algoritmo con el fin de aproximarse a las posibles relaciones que se obtendrían.

Las restricciones impuestas en esta condición han sido que A y B no pueden ser iguales y que la *relation* tiene que ser un verbo, ya que es la acción llevada a cabo por A respecto a B. Cuando se descubre un esquema que cumple los requisitos, este se almacena y se pasa al siguiente párrafo.

Ejecutando esto se obtiene un acercamiento a las relaciones que se esperaban obtener, quedando de la siguiente manera:

```
{'A': 'Almansa', 'B': 'valva', 'relation': 'vieres'}
{'A': 'Malpyca', 'B': 'Lugo', 'relation': 'nacer'}
{'A': 'Briseno', 'B': 'Lugo', 'relation': 'hacer'}
{'A': 'Guadarterme', 'B': 'Lugo', 'relation': 'dar'}
{'A': 'Castellano', 'B': 'cahiz regidor', 'relation': 'dar'}
{'A': 'Almodovar Sabastian Hierro', 'B': 'Mando Castellano Mesa',
'relation': 'dar'}
```

*Problema de los nombres propios de personas

Dado que en el proceso de reconocimiento de entidades los nombres propios no se almacenan como una sola entidad, ha sido necesario solventar esto de forma que [“Juan”, ”Delgado”, ”Almansa”] se convierta en [“Juan Delgado Almansa”], ya que, si no fuera así, no aparecería el nombre completo para las entidades. Esto se ha conseguido así:

I. Cuando se identifica un nombre propio (PROPN) se busca si está precedido por otro PROPN.
II. Cuando hay dos o más PROPN juntos, se almacenan en la misma variable con los mismos datos, salvo el texto que se completa con todos los nombres.

III. Se eliminan los nombres que componen el nombre completo.

Como se puede observar, las relaciones cumplen con la condición establecida, pero los datos de estas en rara ocasión aportan información valiosa. Aun así, se ha decidido continuar para poder elaborar un resultado parecido al que se obtendría generando un modelo del lenguaje que se ajuste mejor utilizado en los textos. Las dificultades nombradas se detallan más profundamente en el apartado 7.1. Problemas y dificultades.

Capítulo 5. Datos y base de datos orientada a grafos.

Uno de los objetivos planteados desde el principio ha sido introducir en una base de datos orientada a grafos (BDOG) los datos sobre las relaciones extraídas de los documentos procesados. Como ya se ha explicado a lo largo de todo el documento, se ha conseguido obtener la información, por lo tanto, ahora el proceso siguiente es incorporar los datos en una BDOG.

Con anterioridad se ha nombrado en la introducción que la herramienta que se va a usar para crear la base de datos es Neo4J, básicamente por ser idónea para almacenar relaciones, que es lo que se buscaba. Para manejar Neo4J tiene su propio lenguaje, por lo que, en este caso, es necesario declarar las sentencias desde Python para indicar a la base de datos lo que tiene que hacer.

5.1. Establecer comunicación entre el entorno Python y Neo4J Aura.

Para entender el funcionamiento es importante distinguir entre dos entornos diferentes, uno el de Python (residiendo en Google Colab) y otro Neo4J (con Neo4J Aura). La comunicación entre ambas tiene lugar gracias a la librería "neo4j", que permite crear una sesión desde Python dentro de la base de datos, tal y como se muestra a continuación:

```
database_connection = GraphDatabase.driver(uri = 'neo4j+s://dc4f3653.databases.neo4j.io',
                                           auth=('xx', 'yy'))
session = database_connection.session()
```

Figura 6. Creación de la sesión con Neo4J Aura.

Una vez se ha creado la sesión, con esta se pueden ejecutar los comandos definidos para la base de datos, solo hay que tener en cuenta a la hora de definirlos que han de ser en formato *string* y tener la sintaxis correcta. Por ejemplo: `'create (A:nodeNameA {name:"Francisco Lopez"}) - [r:relation {name:"hereda"}] -> (B:nodeNameB {name:"Juan Marcos"})'`. Esta sentencia permite crear una relación dentro de la base de datos, por lo tanto, habría que llevar a cabo este proceso para cada relación dentro de los datos obtenidos del procesado del documento. Esto se ha implementado con un bucle que recorra las relaciones, y como previamente se había anticipado esta situación, dentro de cada objeto iterable, se tiene toda la información,

luego se puede ejecutar la sentencia dentro de la sesión con `session.run(_sentencia_).`

El contenido de la base de datos es un acercamiento a cómo sería trabajar con los datos después de llevar a cabo el procesamiento de lenguaje natural.

5.2. Visualización de los datos en Neo4J Aura.

Neo4J Aura [22] es una aplicación en la nube, por lo que para visualizar los datos introducidos por el programa, es necesario acudir a la plataforma online de Neo4J y ahí visualizar los datos dentro de la base de datos (es necesario tener credenciales que se encuentran en el Capítulo 10. Anexo.).

En las siguientes capturas de pantalla se podrá ver como los datos han sido introducidos con éxito y las distintas directrices que hay para visualizar los datos.

Destacar también que toda persona interesada puede acceder a la base de datos mediante la plataforma online desde el enlace que se encuentra en el Capítulo 10. Anexo. de este documento.

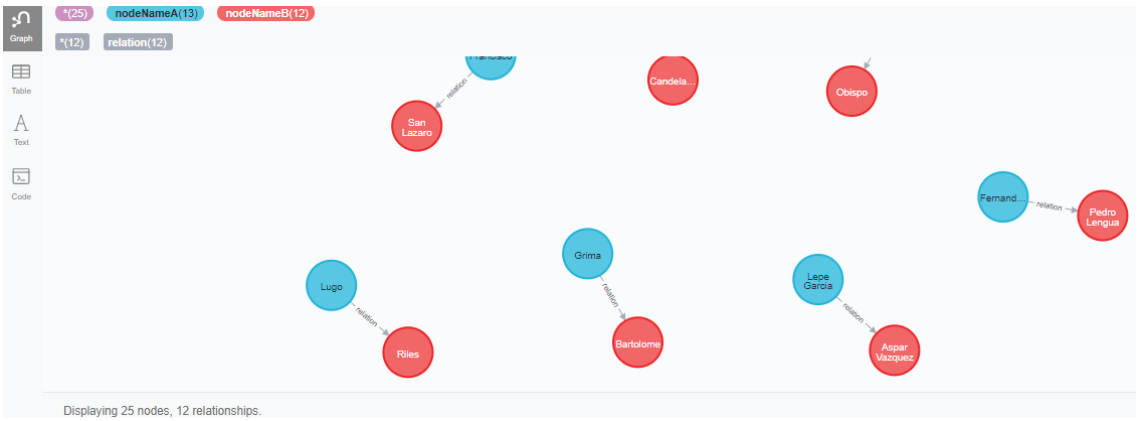


Figura 7. Ejemplo de Neo4J Aura.

Aunque es difícilmente apreciable a través de una captura de pantalla, en la Figura 7. Ejemplo de Neo4J Aura. se puede ver los pares de nodos relacionados con un arco, es decir, los datos han sido introducidos con éxito.

Para visualizar los datos puede procederse de dos maneras:

1. Solicitando la visualización de todo el conjunto de datos con `MATCH (n) RETURN n LIMIT 25` (LIMIT establece el número de nodos que aparecerán, si se omite, aparecen todos los datos)
2. Solicitando la visualización determinando el nodo con `MATCH (n: nodeNameA {name:"Fernando"}) RETURN n.`

Después de obtener un nodo ya se puede ver todos los datos que lo definen, con quién está relacionado, y con qué tipo de relación. Por ejemplo:

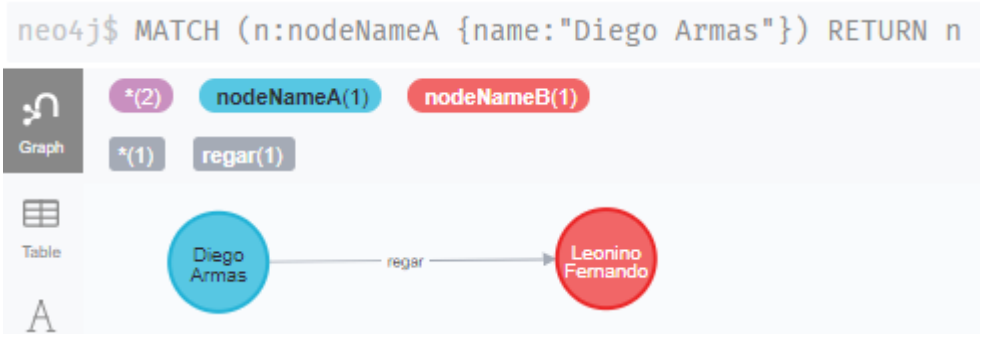


Figura 8. Ejemplo de Neo4J Aura 2.

Capítulo 6. Resultados del proyecto.

Tras haber procesado un texto (*Datas de Tenerife, Libros 1ro al 4to de datas originales*), y de este aproximadamente 350 páginas de información, se ha obtenido en primer lugar, el propio texto depurado y preprocesado sufriendo todos los cambios descritos en el apartado 4.2. Depuración previa del texto. y 4.3. Procesamiento del texto., y en segundo lugar las relaciones de los datos que suponen una parte muy pequeña de toda la información, tal y como se comenta en el apartado anterior. Respecto a lo primero, es sin duda donde se ha concentrado la mayor parte del esfuerzo de este proyecto, y lo segundo se trata más bien de una aproximación al producto esperable de un proyecto de este estilo donde se trabaja con procesamiento de lenguaje natural.

Atendiendo a la parte de los datos de los textos, se ha llegado a etiquetar las palabras del texto y crear un diccionario con información relevante, pero que a la hora de usar este para elaborar relaciones es cuando empezaron a surgir problemas.

En cuanto a las relaciones se buscaba obtener nombres, lugares, fechas, entidades, relaciones, etc., pero al final se ha restringido a nombres propios teniendo una versión limitada de la base de datos. El desarrollo realizado ha permitido generar aproximadamente 600 nodos y 330 relaciones diferentes y ha permitido establecer el flujo de trabajo que permite crearla.

Dentro de Neo4J Aura se pueden ver ciertos datos que determinan que el funcionamiento es correcto y que se podría trabajar sobre esto, tal y como se muestra en la Figura 9. Consulta “lindar” en la BDD.

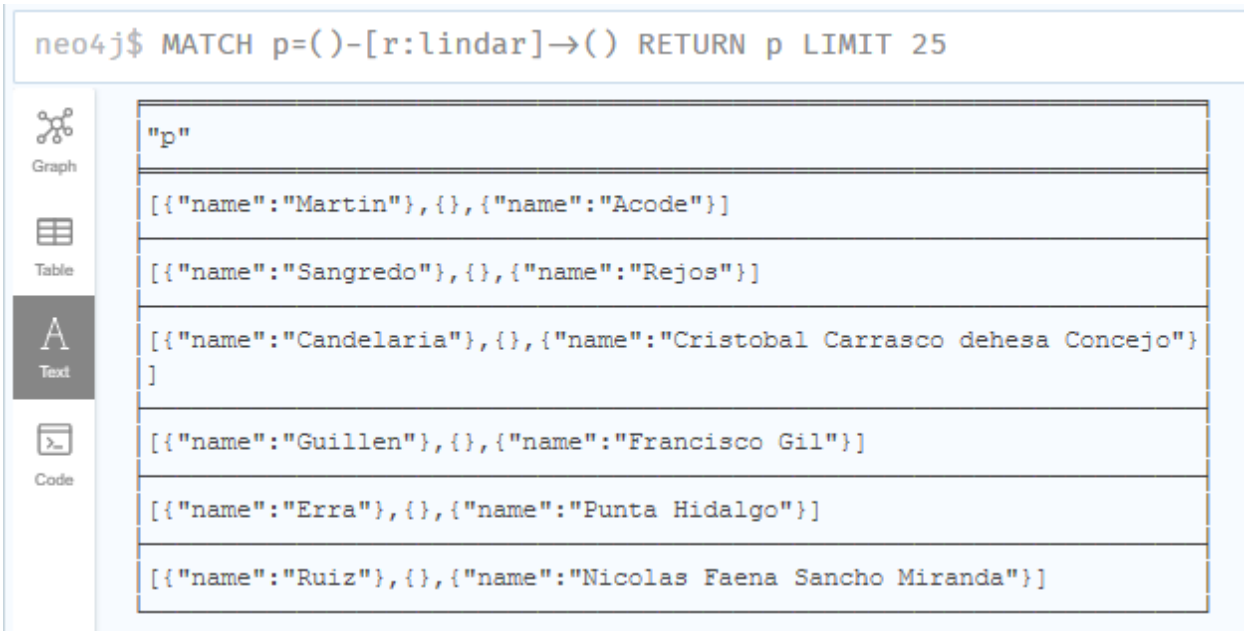


Figura 9. Consulta "lindar" en la BDD.

Finalmente decir que, aunque los resultados del proyecto no hayan sido los esperados, los pasos tomados hasta el final han sido correctos.

Capítulo 7. Conclusiones y futuro del proyecto.

Con todo lo que se ha explicado a lo largo de este documento, ha quedado detallada la implementación del proyecto abarcando todos los aspectos del mismo, desde el primer análisis hasta el resultado final, pasando por todos los procesos propios de procesamiento de lenguaje natural y manejo de bases de datos.

A modo de conclusión para este documento, se van a describir principalmente las dificultades encontradas y el cumplimiento de los objetivos, y luego una valoración personal sobre el trabajo.

7.1. Problemas y dificultades.

Problema	Descripción
Desconocimiento de Python	Antes de comenzar el proyecto, personalmente no había trabajado con el lenguaje de programación de Python, por lo que desde el inicio del trabajo esta ha sido una dificultad notable, puesto que en todo momento se ha tenido que estar recurriendo a la documentación para implementar los algoritmos. Aun con todo, no a supuesto un gran problema, ya que Python posee una curva de aprendizaje muy agradable de cara al usuario.
Confeccionar el mejor entorno de trabajo dentro de las posibilidades	En un principio se optó por trabajar de manera local en la máquina, pero tras muchos conflictos con Python y sus librerías, sumados a Windows y el subentorno de Linux, se buscaron otras opciones, hasta decantarse por un entorno de *cloud computing* como Google Colab, que permitió desarrollar el trabajo con los recursos externos, de una manera mucho más sencilla y sin tener que sufrir contratiempos con la convivencia de las distintas tecnologías encontradas.
Instalación de *freeling* (librería de Python para PLN)	Al comienzo la tutora del proyecto recomendó el uso de *freeling* como principal herramienta de procesamiento de los textos, dada su reputación en este campo, pero surgieron numerosos inconvenientes a la hora de trabajar con esta, pues en la instalación en Python aparecían múltiples errores, y posteriormente en el entorno de Google Colab, por lo que al final se decidió por buscar otras alternativas dentro del ámbito del procesamiento de lenguaje natural.
Formato de los textos objeto de estudio	Importante dificultad encontrada en el desarrollo del proyecto, y ya se ha nombrado a lo largo de esta memoria, y es el formato en el que están los documento a analizar, tanto en el aspecto gramatical como propiamente el correcto uso del lenguaje, aparte de los errores de transcripción. Por ello, hacer frente a esto ha supuesto el mayor esfuerzo y trabajo dentro de este proyecto, véase en el análisis y procesamiento del texto.

	Es por esto que el resultado final del trabajo no es tan satisfactorio como podía anticiparse, y no aporte tanta información como se deseaba.
Errores en las relaciones obtenidas de los datos	Es el mayor problema de todo el TFG, ya que por el hecho de no haber logrado obtener unos datos correctos de los textos, ha dado lugar a no conseguir el objetivo principal del proyecto. Esto ha ocurrido por dos razones, la primera es la incompatibilidad de las tecnologías de procesamiento con el texto analizado, y la segunda, tener el enfoque incorrecto a la hora de desarrollar el reconocimiento de entidades nombradas.
Implementación de algoritmos propios	Por la excepcionalidad y unicidad del objeto de estudio de este trabajo, es decir, los documento a procesar, se ha tenido de dedicar mucho tiempo y esfuerzo a crear algoritmos y funciones para solventar los problemas tales como pulir el texto. Aunque puedan existir herramientas cuyo producto sea el mismo que se buscaba, dado el formato de los documentos, ha sido necesario crear lo mismo, pero para trabajar con estos textos.
Dificultades a nivel personal	A pesar de no haber supuesto un problema grave, la gestión del tiempo, trabajo y constancia para este proyecto han tenido un peso considerable.

Tabla 5. Problemas y dificultades del proyecto.

7.2. Cumplimiento de objetivos.

Tras haber planteado las dificultades en el apartado anterior, ahora es momento de exponer los puntos buenos de este proyecto, que son aquellos objetivos cumplidos, tanto aquellos que se plantearon desde el principio del proyecto, como otros que han surgido durando el desarrollo del mismo. A continuación, se enumeran dichos objetivos:

A. **Trabajar con Python y en un entorno colaborativo de *cloud computing***.
B. **Depurar el texto lo suficiente como para ejecutar un procesamiento de lenguaje natural** y conseguir datos relevantes después de dicho análisis. Luego de todas las dificultades encontradas con los documentos, se consiguió procesar el texto y obtener información para elaborar una base de datos orientada a grafos, la cual usar para enriquecer el conocimiento de los interesados en la historia antigua de las Islas Canarias.
C. **Distinguir las diferentes etapas por las que pasa el texto y obtener su resultado,** en caso de que alguien le interese conocer o usar estos procesos de tratamiento de un texto.

D. **Obtener los datos de la categoría gramatical para el texto procesado**, en este caso sujeto a la denominación aportada por la herramienta spaCy, que es la usada en esta tarea.
E. **Establecer comunicación entre Python** (Google Colab) **y Neo4J Aura** (base de datos orientada a grafo en la nube).
F. **Crear una base de datos orientada a grafo**s con la información extraída de los textos analizados.
G. **Presentar los resultados y confeccionar un informe del proyecto**.

7.3. Líneas futuras del proyecto.

Las dificultades surgidas por la anacronía del texto han sido resueltas en parte en el preprocesamiento y depuración, sin embargo, se espera obtener mejores resultados generando modelos del lenguaje a partir de textos de la época. Estos pueden ser los propios protocolos y acuerdos de cabildo que también se encuentran digitalizados. Atendiendo a la parte de la base de datos, habría que mejorar el manejo de la misma y entender el funcionamiento de la plataforma usada para sacar el máximo provecho de la información almacenada.

7.4. Valoración personal.

Para terminar, mi opinión personal sobre el desarrollo del proyecto es buena, a pesar de no haber conseguido cumplir con las metas esperadas, pero aun así he logrado afrontar numerosas dificultades y llevar el trabajo hasta el final.

Tomé este proyecto a pesar de no tener ningún conocimiento de los lenguajes usados, las tecnologías del campo o los conceptos en los que se basaba, pero aun así puse empeño en aprender, puesto que me parecía un tema interesante. Estoy contento y feliz de haber podido conseguir todo lo que he recogido en este documento.

Chapter 8. Conclusions and future of the project.

With all that has been explained throughout this document, the implementation of the project has been described, covering all aspects of the project, from the first analysis to the final result, including all the processes involved in natural language processing and database management.

As a conclusion to this document, we will describe mainly the difficulties encountered and the achievement of the objectives, and then a personal assessment of the work.

8.1. Problems and difficulties.

Problem	Description
Lack of knowledge of Python	Before starting the project, I personally had not worked with the Python programming language, so from the beginning of the work this has been a notable difficulty, since at all times I had to be resorting to the documentation to implement the algorithms. Even so, it is not a big problem, since Python has a very pleasant learning curve for the user.
To make the best working environment within the possibilities	At the beginning we opted to work locally on the machine, but after many conflicts with Python and its libraries, added to Windows and the Linux sub-environment, we looked for other options, until we opted for a cloud computing environment such as Google Colab, which allowed us to work with external resources in a much simpler way and without having to suffer setbacks with the coexistence of the different technologies found.
Installing freeling (Python library for PLN)	At the beginning, the project tutor recommended the use of freeling as the main text processing tool, given its reputation in this field, but numerous problems arose when working with it, as multiple errors appeared in the Python installation, and later in the Google Colab environment, so in the end it was decided to look for other options within the field of natural language processing.
Format of the texts under study	Possibly this is the biggest problem faced in the development of the project, and it has already been mentioned throughout this report, and it is the format in which the documents to be analyzed are, both in the grammatical aspect and the correct use of language, apart from

	transcription errors. Therefore, dealing with this has meant the greatest effort and work within this project, see in the analysis and processing of the text. This is why the final result of the work is not as satisfactory as could be anticipated, and does not provide as much information as desired.
Errors in the relationships obtained from the data	This is the biggest problem of the whole TFG, since the failure to obtain correct data from the texts has resulted in not achieving the main objective of the project. This has happened for two reasons, the first one is the incompatibility of the processing technologies with the analyzed text, and the second one is having the wrong approach when developing the entity recognition.
Implementation of own algorithms	Due to the exceptionality and uniqueness of the object of study of this work, i.e., the documents to be processed, a lot of time and effort had to be devoted to create algorithms and functions to solve problems such as polishing the text. Although there may be tools whose product is the same as the one, we were looking for, given the format of the documents, it has been necessary to create the same but to work with these texts.
Personal difficulties	Although it was not a serious problem, time management, work and perseverance for this project have had a considerable weight.

Tabla 6. Problems and difficulties of the project.

8.2. Achievement of goals.

After having discussed the difficulties in the previous section, it is now time to present the good points of this project, which are those objectives met, both those that were proposed from the beginning of the project, and others that have appeared during the development of the project. These objectives are listed below:

A. **Work with Python and in a collaborative cloud computing environment**.
B. **Debug the text enough to run natural language processing and get relevant data after such analysis**. After all the difficulties encountered with the documents, it was possible to process the text and obtain the necessary information to elaborate a graph-oriented database to be used to enrich the knowledge of those interested in the ancient history of the Canary Islands.
C. **Distinguish the different stages through which the text passes and to obtain its result** in case someone is interested in knowing or using these text treatment processes.
D. **Obtain the grammatical category data for the processed text,** in this case subject to the name provided by the spaCy tool, which is the one used in this task**.**

E. **Establish communication between Python** (Google Colab) **and Neo4J Aura** (graph-oriented database in the cloud).
F. **Create a graph-oriented database** with the information extracted from the analyzed texts.
G. **Present the results and prepare a project report**.

8.3. Future of the project.

The difficulties arising from the anachronism of the text have been partially resolved in the preprocessing and debugging, however, we hope to obtain better results by generating language models from texts of the time. These could be the protocols and town council agreements themselves, which are also digitized, and the database part of the project needs to be improved and the functioning of the platform used needs to be understood in order to get the most out of the information stored.

8.4. Personal opinion.

To conclude, my personal opinion about the development of the project is good, even though I did not achieve the expected goals, but I still managed to face many difficulties and carry the work to the end.

I took on this project despite not having any knowledge of the languages used, the technologies of the field or the concepts on which it was based, but I still made an effort to learn, since it seemed to me an interesting subject. I am glad and happy to have been able to achieve all that I have collected in this document.

Capítulo 9. Bibliografía.

[1] Proyecto BiographyNet. *Extracting relations between people and events*. http://www.biographynet.nl/

[2] Proyecto Dutch Biography Portal. http://www.biografischportaal.nl/en

[3] Proyecto Memorie di Guerra. http://www.memoriediguerra.it/

[4] Biblioteca virtual Viera y Clavijo. http://www.iecanvieravirtual.org/

[5] Documento de Las datas de Tenerife (1978): libros I a IV de datas originales. http://iecanvieravirtual.org/index.php/catalogo/item/las-datas-de-tenerife-libros-i-a-iv-de-datas-originales.html

[6] Protocolos de Juan Márquez (1518-1521) (Vol. II) (1993). http://iecanvieravirtual.org/index.php/catalogo/item/protocolos-de-juan-marquez-1518-1521-volii.html

[7] Procesamiento de Lenguaje Natural. *Procesamiento de lenguaje natural ¿qué es?* https://www.iic.uam.es/inteligencia/que-es-procesamiento-del-lenguaje-natural/

[8] Part-of-speech. *What is Part of Speech?* https://www.englishclub.com/grammar/parts-of-speech.htm

[9] Reconocimiento de entidades nombradas. https://es.wikipedia.org/wiki/Reconocimiento_de_entidades_nombradas

[10] PLN. *Procesos de PLN*. https://es.wikipedia.org/wiki/Procesamiento_de_lenguajes_naturales#Componentes

[11] Google Colaboratory. *Preguntas frecuentes*. https://research.google.com/colaboratory/faq.html

[12] Python. *Sitio oficial.*
https://www.python.org/

[13] PyPDF. *Librería de Python para trabajar con archivos PDF*.
https://pypi.org/project/PyPDF2/

[14] NLKT. *Sitio oficial de la plataforma Natural Language Toolkit*.
https://www.nltk.org/

[15] Hunspell. *Corrector ortográfico en Python*.
https://unipython.com/hunspell-corrector-ortografico-en-python/

[16] spaCy. *Industrial-Strength Natural Language Processing*.
https://spacy.io/

[17] Bases de Datos Orientadas a Grafos. *¿Qué es una base de datos orientada a grafos?*
https://www.oracle.com/es/big-data/what-is-graph-database/

[18] Neo4J. *Sitio oficial de la plataforma Neo4J para la creación de BDOG*.
https://neo4j.com/

[19] Stopwords. *Definición de stopwords.*
https://es.wikipedia.org/wiki/Palabra_vac%C3%ADa

[20] Lematización. *Definición del proceso de lematización*.
https://es.wikipedia.org/wiki/Lematizaci%C3%B3n

[21] JSON. *Introducción al tipo de dato JSON*.
https://www.json.org/json-es.html

[22] Neo4J Aura. *Cloud Service for Neo4J*.
https://neo4j.com/cloud/aura/

Capítulo 10. Anexo.

- Cuaderno Júpiter (Google Colab) del proyecto aquí.

- Base de datos orientada a grafos (Neo4J Aura) aquí.
  Con las credenciales:
  Usuario:
  neo4j
  Contraseña:
  lXbQHPBHfvYbBN8frYQfpWnF1HeNN10ubGO3EkVGbwg.