

|  |
| --- |
| **Universidad Internacional de La Rioja**  **Escuela Superior de Ingeniería y Tecnología**  **Máster Universitario en Análisis y Visualización de Datos Masivos** |
|  |
| Análisis de la Cronología y relaciones entre personajes bíblicos y eventos a través del tiempo |

**Trabajo Fin de Máster**

**Tipo de trabajo:** Piloto Experimental

**Presentado por:** González Cano, David

**Director:** Fernández García, Diego

**Resumen**

A través del uso de diferentes técnicas de Big Data y Visualización del Procesamiento de Lenguaje Natural (NPL), usando técnicas dirigidas al análisis de modernos análisis de texto, usando Python y herramientas de software libre como Google BERT.

La propuesta de trabajo de Fin de Master (TFM) tiene el propósito central de utilizar estas técnicas para analizar la cronología de los personajes bíblicos, los eventos y las relaciones que estos personajes tienen en los libros de la Biblia.

Para ello se utilizan diferentes datasets públicos que contienen variables como “fecha, “pasaje de la biblia”, mediante técnicas de NPL se analizan las variables y finalmente se crea una plataforma de visualización que sea accesible a cualquier usuario que lo requiera.

Para la recopilación, limpieza, depuración, análisis, clasificación y visualización de datos se utilizará Python con sus respectivas librerías.

Para analizar y trabajar con textos relacionados con la cronología bíblica se va a utilizar **NPL** (Procesamiento de Lenguaje Natural)

Para la visualización se está pensando en herramientas colaborativas que permita obtener tanto visualizaciones estáticas como dinámicas.

Lo que se espera tener de esta investigación es poder dejar un prototipo que utilizando el texto de la biblia permita usar las herramientas para hacer preguntas sobre la cronología tanto de eventos como de personajes y presente visualizaciones claras de lo requerido.

**Palabras Clave:** Procesamiento Natural de Lenguaje, NLP, Cronología Bíblica, Google BERT, GitHub. Google Colab, Visualización, Power Bi, Biblia,

**Abstract**

Through the use of different Big Data and Natural Language Processing (NPL) Visualization techniques, using techniques aimed at the analysis of modern text analysis, using Python and free software tools such as Google BERT.

The Master's Thesis proposal (TFM) has the central purpose of using these techniques to analyze the chronology of biblical characters, the events and the relationships that these characters have in the books of the Bible.

To do this, different public datasets are used that contain variables such as “date,” “Bible passage.” Using NPL techniques, the variables are analyzed and finally a visualization platform is created that is accessible to any user who requires it.

Python with its respective libraries will be used to collect, clean, debug, analyze, classify and visualize data.

To analyze and work with texts related to biblical chronology, NPL (Natural Language Processing) will be used.

For visualization, collaborative tools are being considered that allow obtaining both static and dynamic visualizations.

What is expected from this research is to be able to leave a prototype that, using the text of the Bible, allows the use of tools to ask questions about the chronology of both events and characters and presents clear visualizations of what is required.

**Keywords:** Natural Language Processing, NLP, Biblical Chronology, Google BERT, GitHub. Google Colab, Visualization, Power Bi, Bible,

Índice de Contenido

[1. Introducción 7](#_Toc156172221)

[1.1 Justificación 7](#_Toc156172222)

[1.2 Planteamiento del trabajo 8](#_Toc156172223)

[1.3 Estructura de la memoria 8](#_Toc156172224)

[2. Contexto y estado del arte 9](#_Toc156172225)

[2.1 Introducción al tema 9](#_Toc156172226)

[2.2. Desarrollo (base teórica): antecedentes, estudios actuales, autores de referencia 9](#_Toc156172227)

[2.2.1 Análisis de Texto 10](#_Toc156172228)

[2.2.2 NLP (Natural Lenguaje Processing) 21](#_Toc156172229)

[2.2.2.1 Word Embedding 23](#_Toc156172230)

[2.2.2.2 Pruebas 29](#_Toc156172231)

[2.2.2.3 Haciendo Predicciones 29](#_Toc156172232)

[2.2.3 Google BERT (Bidirectional Encoder Representations from Transformers) to 30](#_Toc156172233)

[2.2.4. Herramientas Colaborativas 31](#_Toc156172234)

[2.2.4.1 GitHub 31](#_Toc156172235)

[*2.2.4.2* Google *Colab* 32](#_Toc156172236)

[2.2.5. Python 32](#_Toc156172237)

[2.2.5.1 Análisis de Texto 33](#_Toc156172238)

[2.2.5.2 Bibliotecas Python 34](#_Toc156172239)

[2.2.5.3 spaCy 35](#_Toc156172240)

[2.2.5.3.1 Tokenización 35](#_Toc156172241)

[2.2.6. La Biblia 62](#_Toc156172242)

[2.2.7. Dataset bíblicas 74](#_Toc156172243)

[2.3 Conclusiones (nexo de unión de lo investigado con el trabajo a realizar) 75](#_Toc156172244)

[3. Objetivos concretos y metodología de trabajo 75](#_Toc156172245)

[3.1. Objetivo general 75](#_Toc156172246)

[3.2. Objetivos específicos 75](#_Toc156172247)

[4. Desarrollo específico de la contribución 75](#_Toc156172248)

[4.1. Descripción detallada del Piloto experimental 75](#_Toc156172249)

[4.1.1 Qué tecnologías se utilizaron (incluyendo justificación de por qué se emplearon y descripciones detalladas de las mismas). 76](#_Toc156172250)

[4.1.2 Cómo se organizó el piloto 76](#_Toc156172251)

[4.1.3 Qué personas participaron (con datos demográficos) o qué técnicas de evaluación automática se emplearon. 77](#_Toc156172252)

[4.1.4. Cómo transcurrió el experimento. 78](#_Toc156172253)

[4.1.5 Qué instrumentos de seguimiento y evaluación se utilizaron. 78](#_Toc156172254)

[4.2 Descripción de los resultados 78](#_Toc156172255)

[4.3. Discusión 80](#_Toc156172256)

[5. Conclusiones y trabajo futuro 81](#_Toc156172257)

[5.1. Conclusiones 81](#_Toc156172258)

[5.2. Líneas de trabajo futuro 81](#_Toc156172259)

[Bibliografía 83](#_Toc156172260)

[Anexos 87](#_Toc156172261)

[Anexo I. Entrevista 1 87](#_Toc156172262)

[Anexo 2. Código 89](#_Toc156172263)

[Anexo 3. Código 90](#_Toc156172264)

[*Federación Bautista independiente de Colombia.* 90](#_Toc156172265)

**Índice de figuras**

[Figura 1 Taxonomía del machine Learning dentro de la Inteligencia Artificial. Basado en Panesar, 2019 10](#_Toc156172323)

[Figura 2 Tasa de crecimiento de los datos entre 2006-2020 (Cheesem, n.d.) 11](#_Toc156172324)

[Figura 3 Un ejemplo de un modelo de traducción neuronal, trabajando del francés al inglés. (Introduction to Neural Machine Translation with GPUs (Part 2), 2015) 14](#_Toc156172325)

[Figura 4 Técnicas como el modelado de temas utilizan métodos de modelado probabilístico para identificar temas clave del texto. (Thiyagarajan, 2018) 15](#_Toc156172326)

[Figura 5 Vuelve a golpear el clavo con el martillo (A webcomic of romance,, n.d.) 19](#_Toc156172327)

[Figura 6 Un fragmento de una proyección de 2D 26](#_Toc156172328)

[Figura 7 Generando un modelo estadístico con el algoritmo de aprendizaje automático 29](#_Toc156172329)

[Figura 8 Ejemplo Predicción una estructura de árbol de dependencia usando un modelo estadístico 30](#_Toc156172330)

[Figura 9 Vistazo General de los pasos de NLP 35](#_Toc156172331)

[Figura 10 Ejemplo Tokenización 36](#_Toc156172332)

[Figura 11 Parte de la visualización de cronología de la Biblia (Aschmann, 2022) 74](#_Toc156172333)

**Tabla de Contenido**

[Tabla 1 Simplificado Espacio vectorial de palabras 24](#_Toc156172342)

[Tabla 2 Operación matemática vectorial en un espacio vectorial de palabras 25](#_Toc156172343)

[Tabla 3 Libros de la Biblia Hebrea o Antiguo testamento 64](#_Toc156172344)

[Tabla 4 Libros Antiguo testamento mayoría biblias cristianas 65](#_Toc156172345)

[Tabla 5 Antiguo testamento (Antiguo Testamento, n.d.) 70](#_Toc156172346)

[Tabla 6 Libros del nuevo testamento más aceptaods (Nuevo Testamento, n.d.) 73](#_Toc156172347)

# 1. Introducción

## 1.1 Justificación

Meditando sobre los proyectos que dan gasolina en el día a día y que se vuelven retos ultraístas, el leer la Biblia es uno de ellos, desde su inicio hasta el final contando con más de cuarenta escritores y aproximadamente 773.746 palabras en un libro que nos lleva desde el inicio del mundo, va recorriendo con eventos, personajes los diferentes tiempos de la humanidad y ahí, en el mismo texto lleno de aventuras nos sumergimos en cual será el final de la humanidad con señales que se conectan en los 66 libros divididos en 1189 capítulos y 31.102 versículos y 3.5 35.666.480 letras, Biblia Reina Valero 1960. (Concepto y definicion Net, 2023)

La secuencia de los tiempos, los personajes, los eventos en los escritos no son tan fáciles de visualizar debido a que las fechas no están implícitas en el texto y se necesitan estudios teológicos y contextos históricos para poder entender la cronología de los eventos con sus respectivos personajes.

Para llevar a cabo esta investigación se van a utilizar diferentes técnicas, métodos y herramientas para encontrar una forma de analizar y presentar cronológicamente eventos, personajes utilizando textos que están ya colocados en la web y también si es posible desarrollar un prototipo de base de datos que nos permita ir actualizando las fechas en una línea de tiempo actualizable por los investigadores de este tema.

La necesidad de esta investigación surge de la diversidad de autores en sus escritos, las figuras literarias y estilísticas que se utilizan como metáforas, símiles, personificaciones, parábolas, hipérboles, ironías, y muchas más; la identificación de fechas en cada evento, la no existencia hasta el momento de una base de datos estructurada que coloque cada evento o personaje en una línea de tiempo.

Por todo ello se propone una herramienta a la población interesada en poder dar un contexto en el tiempo de los escritos ya sea por estudio u otro interés dando una visualización en una forma clara y utilizando herramientas de datos masivos.

En la población mundial aproximadamente de 8300 millones y con un cálculo de 2500 millones de cristianos en el mundo (EL orden Mundial EOM, 2019), tienen como base de su fe la Biblia, la cual se divide en Antiguo y nuevo testamento. Así a esta población se puede sumar el pueblo judío aproximadamente 13 millones (Embajada de Israel en España, s.f.), los cuales su base de lectura en su creencia es el Antiguo Testamento. ya que tienen una división diferente y se conoce como la biblia hebrea o Tanaj la cual está dividida en tres partes: la Torá (ley) , los Nevi’im (profetas) y los Ketuvim (escritos), además los judíos no consideran la parte de los libros del nuevo testamento como parte de las escrituras.

## 1.2 Planteamiento del trabajo

Utilizando técnicas de procesamiento de lenguaje natural (NLP) y modelos de machine Learning que permitan extraer información y procesarla cronológicamente. Creando una base de datos estructurada que manaje y permita trabajar con fechas de los personajes bíblicos y utilizar herramientas de visualización que permitan de una forma fácil presentar los eventos y personajes en forma dinámica.

## 1.3 Estructura de la memoria

La primera parte es describir las técnicas de manejo de Texto

Lo segunda parte está dirigida al análisis de la biblia como un dataset y como se logra organizar para obtener información cronológica

El tercer es el prototipo planteado de la unión de las técnicas y la estructura bíblica con herramientas de Visualización.

# 2. Contexto y estado del arte

## 2.1 Introducción al tema

Es interesante pensar que cuando hablamos de tiempo, aunque no lo sentimos material, tiene estados de pasado, presente y futuro; y cuando lo graficamos con personajes y eventos se nos convierte en una línea de tiempo, una herramienta visual para presentar cualquier información que mezcla datos y eventos formando una función de sucesión temporal.

La cronología bíblica se refiere al estudio y ordenamiento de los eventos históricos descritos en la biblia en secuencia temporal y busca determinar fechas aproximadas de eventos, así como la duración de periodos y la secuencia en la que ocurrieron.

El análisis de texto mediante el procesamiento del Lenguaje natural aplicado a el texto bíblico tiene varios temas de interés, así como la identificación de personajes, eventos y sus relaciones en el tiempo usando diferentes técnicas.

## 2.2. Desarrollo (base teórica): antecedentes, estudios actuales, autores de referencia

El proyecto se enmarca en la Disciplina de la inteligencia artificial que, según la definición de la real academia española, Inteligencia Artificial: “es una Disciplina científica que se ocupa de crear programas informáticos que ejecutan operaciones comparables a las que realiza la mente humana, como el aprendizaje o el razonamiento lógico” (ASALE, n.d.)

Como un subcampo encontramos el procesamiento de lenguaje natural que trata el procesar y analizar los datos de lenguaje natural. Esto incluye enseñarles a máquinas para interactuar con humanos con un lenguaje que se desarrolla naturalmente a través del uso, creando algoritmos de Machine Learning que trabajan con grandes datasets desconocidas. Hoy en día se pueden usar algoritmos para observar cuyas reglas semánticas y gramaticales son bien conocidas. No es sorprendente que generar y comprender el lenguaje natural sea la tarea más prometedora y, al mismo tiempo, más desafiante, involucrada en NPL. (Vasiliev, 2020)

.

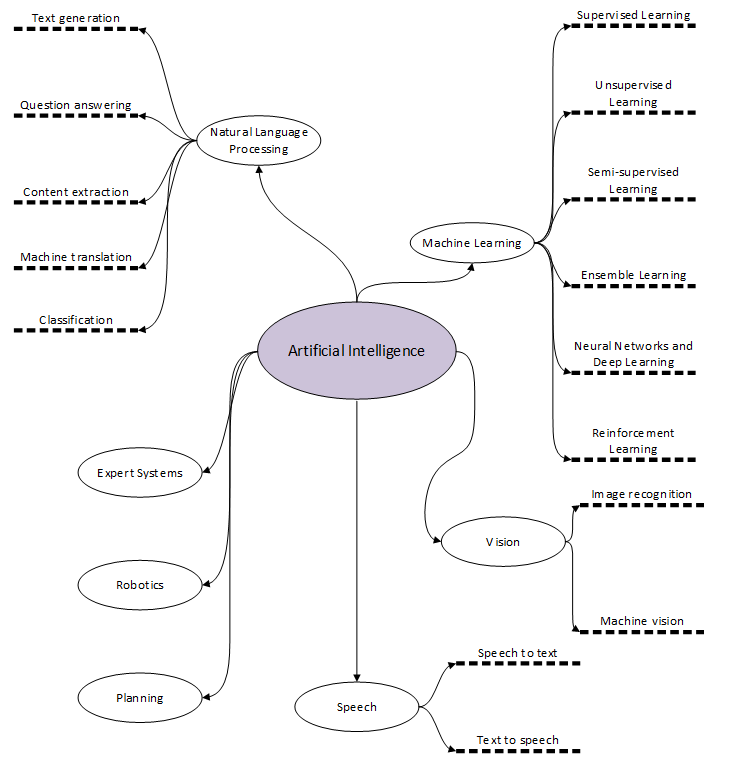


Figura 1 Taxonomía del machine Learning dentro de la Inteligencia Artificial. Basado en Panesar, 2019

.

## 2.2.1 Análisis de Texto

(Srinivasa-Desikan, 2018)Actualmente el análisis de texto se puede realizar usando Python y otras herramientas abiertas que trabajan con algoritmos que están disponibles como spaCy y Gensim.

Si hay un medio de comunicación que en mundo está inmerso, es el **texto**. Desde los periódicos, revistas, internet, email, mensaje de texto y la mayoría de información que se recibe. La cantidad de información que se maneja en compañías como Google (más de un billón de consultas por año) Twitter (1.6 billones de mensajes por día), y WhatsApp (30 mil millones de mensajes por día). Los datos textuales tienen un enorme valor comercial y las empresas pueden utilizar estos datos para ayudar a perfilar los clientes y comprender tendencias. Esto puede utilizarse para ofrecer una experiencia más personalizada a los usuarios o como información de mercadeo dirigido. Facebook, por ejemplo, utiliza una gran cantidad de datos textuales, y uno de los algoritmos fue desarrollado por el equipo de AI de Facebook. (Srinivasa-Desikan, 2018)

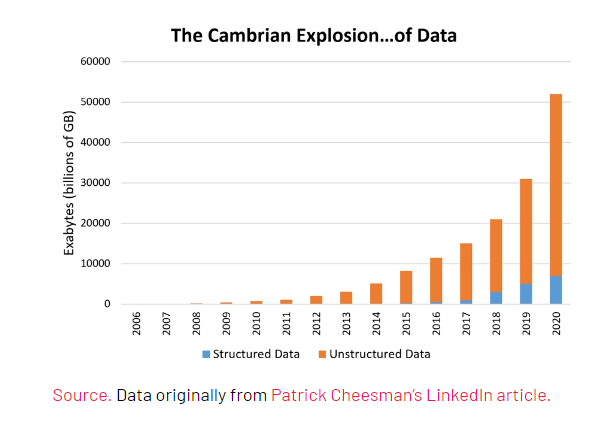


Figura 2 Tasa de crecimiento de los datos entre 2006-2020 (Cheesem, n.d.)

El análisis de texto puede entenderse como la técnica de extraer información útil del mismo texto, lo cual se puede hacerse mediante varias técnicas ya sea utilizando el procesamiento del lenguaje natural (NLP), Lingüística Computacional (CL) y herramientas numéricas para obtener esta información. Estas herramientas numéricas son algoritmos de aprendizaje automático o algoritmos de recuperación de información.

El procesamiento del lenguaje natural (NLP) se refiere al uso de una computadora para procesar el lenguaje natural. Por ejemplo, eliminar todas las apariciones de la palabra de un cuerpo de texto es un ejemplo de ello, aunque sea un ejemplo básico.

La lingüística computacional (CL), como su nombre indica, es el estudio de la lingüística desde una perspectiva computacional. Esto significa utilizar computadoras y algoritmos para realizar tareas lingüísticas, como marcar el texto como parte del discurso (como sustantivo o verbo), en lugar de realizar esta tarea manualmente.

Machine Learning (ML) es el campo de estudio en el que utilizamos algoritmos estadísticos para enseñar a las máquinas a realizar una tarea particular. Este aprendizaje ocurre con datos y nuestra tarea a menudo es predecir un nuevo valor basado en datos observados previamente.

La recuperación de información (IR por sus siglas en inglés, Information Retrieval ) es la tarea de buscar o recuperar información en función de una consulta del usuario. Los algoritmos que ayudan a realizar esta tarea se denominan algoritmos de recuperación de información e incluyen temas como indexación, modelos de recuperación, relevancia y evaluación que mejoran la calidad de los resultados de las búsquedas y la interacción entre Usuario-Sistema que mejora la precisión de las búsquedas. Es importante destacar en este punto que la precisión se refiere a la proporción de los documentos que son relevantes. Este punto es muy común en motores de búsqueda, bibliotecas digitales donde la recuperación de documentos y búsqueda de información textual son fundamentales.

El análisis de texto en sí existe desde hace mucho tiempo: una de las primeras definiciones de Business Intelligence (BI), en un artículo del IBM Journal de octubre de 1958 escrito por H. P. Luhn, A Business Intelligence System (L-uhn, 1958) , describe un sistema que hará el siguiente:

"...utilizar máquinas de procesamiento de datos para la abstracción y codificación automática de documentos y para crear perfiles de interés para cada uno de los 'puntos de acción' de una organización. Tanto los documentos entrantes como los generados internamente se abstraen automáticamente, caracterizándose por una palabra patrón y se envía automáticamente a los puntos de acción apropiados".

Es interesante hablar acerca de documentos, en lugar de números; pensar que las primeras ideas de la inteligencia empresarial fueron comprender texto y documentos es testimonio del análisis de texto a lo largo de los siglos. Pero incluso fuera del ámbito del análisis de texto para empresas, el uso de computadoras para comprender mejor el texto y el lenguaje ha existido desde el comienzo de las ideas de inteligencia artificial. La revisión de 1999 sobre el análisis de textos realizada por John Hutchins, Retrospect and prospect in computer-based Translation (Hutchins, 1999) habla de los esfuerzos por realizar traducción automática ya en la década de 1950 por parte del ejército de los Estados Unidos, con el fin de traducir revistas científicas rusas al inglés.

Los esfuerzos para crear una máquina inteligente también comenzaron con texto: el programa ELIZA desarrollado en 1966 en el MIT por Joseph Weizenbaum es un ejemplo. Aunque el programa no tenía una comprensión real del lenguaje, mediante la coincidencia de patrones básicos podía intentar mantener una conversación. Estos son sólo algunos de los primeros intentos de analizar texto.

La traducción automática en sí ha recorrido un largo camino y ahora se puede usar teléfonos inteligentes para traducir de manera efectiva entre idiomas, y con técnicas de vanguardia como Google’s Neural Machine Translation, la brecha entre la academia y la industria se está reduciendo permitiendo experimentar de primera mano la magia del procesamiento del lenguaje natural.

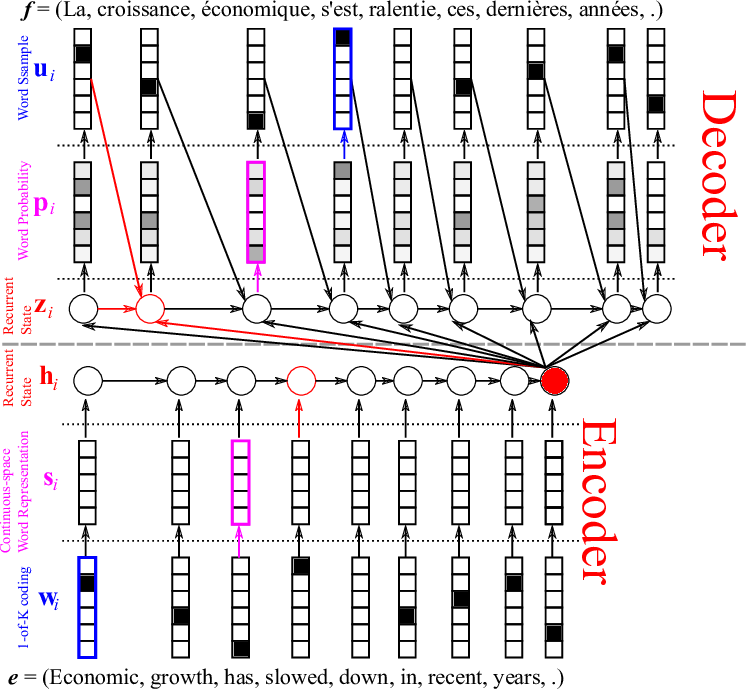


Figura 3 Un ejemplo de un modelo de traducción neuronal, trabajando del francés al inglés. (Introduction to Neural Machine Translation with GPUs (Part 2), 2015)

Los avances en este tema han ayudado a mejorar la forma en que se aborda el habla y los subtítulos en videos, los asistentes personales como Siri de Apple o Alexa de Amazon se benefician enormemente de un procesamiento de texto superior. Comprender la estructura de las conversaciones y extraer información fueron problemas clave en los inicios de la NLP, y los frutos de la investigación están siendo muy evidentes en el siglo XXI.

Los motores de búsqueda como Google o Bing también se apoyan en las investigaciones realizadas en NLP y CL y afectan la forma de vivir de una manera sin precedentes. La recuperación de información se basa en enfoques estadísticos en el procesamiento de textos y nos permite clasificar, agrupar y recuperar documentos. Métodos como el modelado de temas pueden ayudar a identificar temas clave en cuerpos de texto grandes y no estructurados. Identificar estos temas va más allá de la búsqueda de palabras clave y utilizar modelos estadísticos para comprender mejor la naturaleza subyacente de los cuerpos de texto. Sin el poder de las computadoras, no se puede realizar este tipo de análisis estadístico a gran escala del texto.

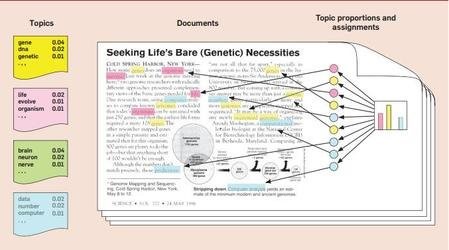


Figura 4 Técnicas como el modelado de temas utilizan métodos de modelado probabilístico para identificar temas clave del texto. (Thiyagarajan, 2018)

El experimentar la informática moderna en los teléfonos móviles, los recientes desarrollos tanto en Python como en NPL significan que ahora se puede desarrollar dichos sistemas por cuenta propia. No sólo ha habido una evolución en las técnicas utilizadas en PNL y análisis de texto, sino que también se vuelven muy accesibles para todos: los paquetes de código abierto se están convirtiendo en herramientas de última generación y con un rendimiento tan bueno como comercial. Un ejemplo de herramienta comercial sería la API de análisis de texto de Microsoft. (Azure AI Language, n.d.)

MATLAB es otro ejemplo de una herramienta comercial popular utilizada para la informática científica. Si bien históricamente este tipo de herramientas comerciales funcionaron mejor que el software gratuito de código abierto, un aumento en el número de personas que contribuyen a las bibliotecas de código abierto, así como la financiación de la industria, han ayudado enormemente a la comunidad de código abierto. Ahora, las cosas parecen haber cambiado y muchos gigantes del software utilizan paquetes de código abierto para sus sistemas internos, como Google que utiliza TensorFlow y Apple que utiliza scikit-learn. TensorFlow y scikit-learn son dos paquetes de aprendizaje automático de Python de código abierto.

Se puede argumentar que la gran cantidad de paquetes que ofrece el ecosistema Python significa que lidera el grupo cuando se trata de realizar análisis de texto. Los proyectos en comunidades de código abierto son una fortaleza cuando se trabaja en investigaciones que involucran muchas tecnología y personas con diferentes profesiones de investigación y aplicación tanto en el mundo laboral. (Srinivasa-Desikan, 2018)

**Como acceder a los datos.**

Si bien es importante conocer las técnicas y herramientas involucradas en PNL y CL, por supuesto, es inútil sin ningún dato. Por suerte para nosotros, tenemos acceso a una gran cantidad de datos si buscamos en los lugares correctos. Una forma de encontrar datos textuales con los que trabajar es buscar un corpus.

Un corpus de texto es un conjunto grande y estructurado de textos y es una excelente manera de comenzar con el texto. análisis. Ejemplos de corpus libres son el Open American National Corpus (CORPUS, n.d.) o el British National Corpus (The British National Corpus (BNC), n.d.). Wikipedia tiene una lista útil de los corpus más grandes disponibles en su artículo sobre corpus de texto (List of text corpora, 2023). Estos no se limitan al idioma inglés, también existen varios corpus en idiomas europeos y asiáticos, y hay esfuerzos constantes en todo el mundo para crear corpus para la mayoría de los idiomas. Los laboratorios de investigación de las universidades son otra fuente valiosa para obtener corpus; de hecho, uno de los corpus más emblemáticos del idioma inglés, el Brown Corpus, se elaboró ​​en la Universidad de Brown.

Existen corpus que se han creado específicamente para representar textos bíblicos en el ámbito del procesamiento de lenguaje natural (NLP) y la lingüística computacional. Estos corpus son colecciones estructuradas de textos bíblicos que se utilizan para realizar análisis lingüísticos, entrenar modelos de lenguaje y llevar a cabo investigaciones relacionadas con la lingüística y la interpretación de textos religiosos.

Algunos ejemplos son Society of Biblical Literature Greek New Testament, un corpus que representa el nuevo testamento griego junto con información como lemas y etiquetas morfológicas. (SBLGNT, n.d.), WLC (Westminster Leningrad Codex) Un corpus representa el antiguo testamento en Hebreo (Center, n.d.), BibleCorpus un corpus que incluye el texto completo de la biblia en muchos idiomas y traducciones que puede ser utilizado para aplicaciones NLP.

Según la estructura y los distintos niveles de información presentes en los corpus, tendría un propósito diferente. Algunos corpus también se crean para evaluar tareas de agrupación o clasificación, donde en lugar de que la anotación sea importante, lo sería la etiqueta o la clase. Esto significa que algunos corpus están diseñados para ayudar con tareas de aprendizaje automático, como agrupaciones o clasificación, proporcionando texto con etiquetas etiquetadas por humanos. La agrupación se refiere a la tarea de agrupar objetos similares, y la clasificación es el proceso de decidir qué clase predefinida e identificar para qué se utilizará exactamente su conjunto de datos es una parte crucial del análisis de texto y un primer paso importante.

Además de descargar conjuntos de datos o extraer datos de Internet, todavía existen algunas fuentes ricas para recopilar nuestros datos textuales, en particular, literatura. Un ejemplo de ello es la investigación realizada en la Universidad de Pensilvania, donde Alejandro Ribeiro, Santiago Segarra, Mark Eisen y Gabriel Egan descubrieron posibles colaboradores de Shakespeare, un problema de historia literaria que hizo tropezar a muchos investigadores (Today, 2023) . Abordaron el problema identificando estilos literarios, un campo de estudio de próxima aparición en lingüística computacional llamado análisis de estilo.

El mayor uso de herramientas computacionales para realizar investigaciones en humanidades también ha llevado al crecimiento de los laboratorios de Humanidades Digitales en las universidades, donde los enfoques de investigación tradicionales se ven favorecidos o superados por la informática y, en particular, el aprendizaje automático (y por extensión), la ciencia natural. procesamiento del lenguaje. Los discursos de los políticos o los procedimientos en el parlamento, por ejemplo, son otro ejemplo de fuente de datos que se utiliza con frecuencia en esta comunidad. They WorkFor You (mySociety, n.d.) es un sistema de seguimiento del parlamento del Reino Unido, que recibe discursos y los carga y es un ejemplo de los muchos sitios disponibles que realizan este tipo de trabajo.

El Proyecto Gutenberg es probablemente el mejor recurso para descargar libros y contiene más de 50.000 libros electrónicos gratuitos y muchos clásicos literarios. Los archivos PDF y libros electrónicos personales también siguen siendo un recurso, pero nuevamente, es importante conocer la naturaleza legal de su texto antes de analizarlo.

Descargar una copia pirateada de, digamos, Harry Potter de Internet y publicar los resultados del análisis de texto puede no ser la mejor idea si no puedes explicar de dónde sacaste el texto, el análisis de texto en mensajes de texto privados infringe las leyes de privacidad.

Hay texto en Internet que se puede acceder (List of text corpora, n.d.) es un ejemplo de ello, y el volcado multimedia de todo el contenido en Wikipedia, después de descomprimirlo, es alrededor de 58 GB (a abril de 2018): texto más que suficiente para jugar. El popular sitio web de agregación de noticias reddit.com (reddit , n.d.)permite un fácil raspado web y es otro gran recurso para el análisis de texto.

Python nuevamente sigue siendo una excelente opción para este tipo de web scraping, y bibliotecas como BeautifulSoup (Python library for pulling data out of HTML and XML files, n.d.), urllib (Python, n.d.) y scrapy (Scrapy, n.d.) están diseñadas especialmente para esto. Es importante tener cuidado con el aspecto legal de las cosas aquí y asegurarse de verificar los términos y condiciones del sitio web del que está extrayendo los datos; varios sitios web no le permitirán utilizar la información del sitio web.

Twitter o el nuevo X es otro sitio web de análisis de texto; se han creado herramientas completas (Social Media Sentiment Visualization, n.d.) para realizar análisis de sentimientos. La API de transmisión de Twitter también nos permite extraer fácilmente datos textuales de Twitter, y la interfaz de Python (Tweepy, n.d.). La mayoría de los líderes mundiales son usuarios de Twitter, así como celebridades y grandes corporaciones de noticias; Twitter puede ofrecernos muchas ideas interesantes.

Otros ejemplos de información textualde Internet incluyen artículos de investigación, informes médicos, reseñas de restaurantes ( conjunto de datos de Yelp!) y otros sitios web de redes sociales. El análisis de sentimientos suele ser el objetivo principal en estos casos. Como sugiere el nombre, el análisis de sentimientos se refiere a la tarea de identificar sentimientos en el texto. Estos sentimientos pueden ser básicos, como sentimientos positivos o negativos, pero podríamos tener tareas de análisis de sentimientos más complejas en las que analicemos si una oración contiene sentimientos felices, tristes o enojados.

Enviamos y recibimos mensajes de texto y correos electrónicos todos los días, y se puede usar este texto para el análisis de texto. La mayoría de las aplicaciones de mensajería de texto tienen interfaces para descargar chats. WhatsApp, por ejemplo, le enviará los datos por correo (Whatsapp Preguntas, n.d.), tanto con medios como con texto. La mayoría de los clientes de correo tienen la misma opción, y la ventaja en ambos casos es que este tipo de datos suele estar bien organizado, lo que permite una fácil limpieza y preprocesamiento antes de sumergirnos en los datos.

Un aspecto que hemos ignorado hasta ahora al hablar de datos es el ruido que a menudo hay en el texto: en los tweets, por ejemplo, los formularios cortos y los emoticones que se utilizan con frecuencia y, en algunos casos, tenemos datos multilingües en los que un análisis simple podría fallar. Esto nos lleva al aspecto posiblemente más importante del análisis de texto: el preprocesamiento.

**Basura dentro basura fuera**

Basura entra, basura sale (o GIGO) es un adagio de la informática que es aún más importante cuando se trata de aprendizaje automático y posiblemente aún más cuando se trata de datos textuales. Basura entra, basura sale significa que si tenemos datos mal formateados,

Es probable que tengamos malos resultados.

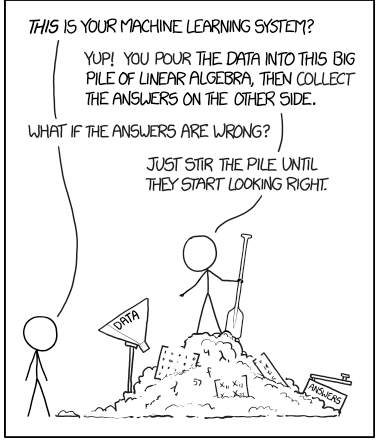


Figura 5 Vuelve a golpear el clavo con el martillo (A webcomic of romance,, n.d.)

Si bien más datos generalmente conducen a una mejor predicción, no siempre es el mismo caso con el análisis de texto, donde más datos pueden generar resultados sin sentido o resultados que no siempre queremos. Un ejemplo intuitivo: las partes del discurso, los artículos, como las palabras que tienden a aparecer mucho en el texto, pero no agregan ninguna información al texto y generalmente se limitan a la gramática o la estructura.

Palabras como estas que no proporcionan información útil se denominan palabras vacías y, a menudo, estas palabras se eliminan del texto antes de aplicarles técnicas de análisis de texto. De manera similar, a veces eliminamos palabras con muy alta frecuencia en el cuerpo del texto y palabras que solo aparecen una o dos veces; es muy probable que estas palabras no sean útiles para nuestro análisis. Dicho esto, esto depende en gran medida del tipo de tarea que se realiza: si, por ejemplo, quisiéramos replicar los estilos de escritura humanos, las palabras vacías son importantes porque los humanos usan muchas de esas palabras cuando escriben. Un ejemplo de cómo las palabras vacías también pueden incluir información útil se encuentra en este artículo, Detección de pastiche basada en clasificaciones de palabras vacías. Un estudio que expuso a imitadores de un escritor rumano (Liviu P. Dinu, 2012) identificó a un determinado autor utilizando la frecuencia de palabras vacías

Otro ejemplo de datos inútiles es buscar palabras o temas influyentes en el texto que al quitarlas no provocaría ninguna pérdida de información. Pero en una nota similar, tendría sentido que las palabras información e informar existieran por separado en el mismo cuerpo de texto, porque podrían significar cosas diferentes según el contexto. Entonces necesitaríamos técnicas para acortar las palabras de forma adecuada. Lematizar y derivar son dos métodos que utilizamos para abordar este problema y siguen siendo dos de los conceptos centrales en el procesamiento del lenguaje natural.

Incluso después de un procesamiento de texto básico, los datos siguen siendo una colección de palabras. Dado que las máquinas no comprenden inherentemente los conceptos vinculados a las palabras, se puede usar números que representen palabras individuales. El siguiente paso importante en el análisis de texto es convertir palabras en números, ya sea una bolsa de palabras (BOW) o una frecuencia de documento inversa de términos (TF-IDF), que son diferentes formas de contar el número de palabras en cada documento o sentencia. También existen técnicas más avanzadas para representar palabras como Word2Vec y GloVe.

**Por qué hacer análisis de texto**

La abundancia de datos fácilmente disponibles que se puede utilizar en la era de big Data nos permite revisar lo que realmente significan todos nuestros datos. De hecho, además de los conjuntos de datos masivos que se pueden descargar de Internet, también el tener acceso a datos pequeños: mensajes de texto, correos electrónicos y una colección de poemas, son algunos ejemplos. Los datos textuales son aún más fáciles de interpretar y comprender los resultados del análisis. Es posible que los números no siempre tengan sentido y no siempre sean atractivos a la vista, pero las palabras son más fáciles de comprender.

El análisis de texto es interesante también porque se puede utilizar datos que involucran directamente al usuario: las conversaciones de texto, los libros, los tweets de personajes o celebridades. La naturaleza personal de los datos de texto siempre añade un poco más de motivación y probablemente también signifique que hace que se tome consciencia de la naturaleza de los datos y del tipo de resultados que se pueden esperar.

Las técnicas de NPL también pueden ayudarnos a construir herramientas que puedan ayudar a negocios o empresas personales: los chatbots, por ejemplo, se están volviendo cada vez más comunes en los principales sitios web y, con el enfoque correcto, es posible tener un chatbot personal. Esto se debe en gran medida a un subcampo del aprendizaje automático, llamado Deep Learning, donde utilizamos algoritmos y estructuras que se inspiran en la estructura del cerebro humano. Estos algoritmos y estructuras también se denominan redes neuronales. Los avances en el aprendizaje profundo han introducido poderosas redes neuronales, como las redes neuronales recurrentes (RNN) y las redes neuronales convolucionales (CNN). Ahora, incluso con un conocimiento mínimo del funcionamiento matemático de estos algoritmos, las API de alto nivel nos permiten utilizar estas herramientas. Integrar esto en nuestra vida diaria ya no está reservado a los investigadores en informática o a los ingenieros a tiempo completo; con la recopilación adecuada de datos y paquetes de código abierto, esto está dentro de nuestras capacidades. Paquetes de código abierto se han convertido en un estándar de la industria: Google ha lanzado y mantiene TensorFlow (TensorFlow, n.d.), y Apple y Spotify utilizan paquetes como scikit-learn (David Cournapeau, n.d.), y spaCy (RoBERTa, 2020).

Ya no estamos limitados ni por los datos ni por las herramientas, las dos únicas cosas que necesitaríamos para realizar un análisis de texto.

Lenguaje de programación Python y las herramientas que utilizaremos serán todas software gratuito de código abierto. En el mundo de la investigación, el código fuente abierto significa que los resultados académicos son reproducibles y están disponibles para todos los interesados. Python sigue siendo un lenguaje potente y fácil de usar y sirve como una excelente manera de ingresar al mundo del procesamiento del lenguaje natural.

### 2.2.2 NLP (Natural Lenguaje Processing)

“Un idioma no son solo palabras. Es una cultura, una tradición, una unificación de una comunidad, una historia completa que cruza lo que es una comunidad. Todo está incorporado en un lenguaje".

-Noam Chomsky

Es un campo entre la intersección de ciencias de computación e inteligencia artificial y logística. Es un tema de construir sistemas que procesan y entienden el lenguaje humano.

(Sowmya Vajjala, 2020)

El lenguaje natural es el primer medio de comunicación entre humanos desde los comienzos de la humanidad. Los computadores siendo objetos no emocionales se comunican en lenguaje de máquina, una codificación de 0s y 1s. La forma en que las maquinas puede entender el lenguaje es lo que se denomina Procesamiento de lenguaje natural.

Lo primero que hay que decir es que tenemos que traducir el lenguaje natural de palabras o texto a números.

El término que se maneja para la representación vectorial de los datos es “embedding” y se utiliza para diversos campos como el NPL, la recomendación de contenido y la visión por computadora. Los algoritmos comunes utilizados en diferentes contestos son

1. Word Embedding
   * Word2Vec: Desarrollado por Google, utiliza una red neuronal para aprender representaciones vectoriales de palabras
   * GloVe (Global Vector for Word Representación) Utiliza estadísticas globales de coocurrencia para generar embeddings
2. Documento Embeddings
   * Doc2Vec: Una extensión de Word2vec que aprende representaciones vectoriales para documentos.
3. Embeddings para imágenes
   * CNN (Convulutional Neural Network) Embeddings: Las capas de intermedias de una CNN pueden utilizarse como embeddings para imágenes
   * Siamese Networks: Se utilizan para aprender embeddings que representa similitudes y diferencias entre pares de imágenes.
4. Embeddings para Datos Tabulares
   * FastText: Se utiliza no solo para palabras, sino también para representar datos tabulares.
   * Entity Embeddings of Categorical Variables: Se emplea en conjuntos de datos con variables categóricas.
5. Graph Embeddings
   * Node2Vec: Extiende el concepto de Word2Vec para grafos, generando embeddings para nodos.
   * GraphSAGE (Graph Sample and Aggregated): Genera embeddings para nodos agregando información de sus vecinos.
6. Embeddings para secuencias temporales
   * LSTM (Long Short-term memory) Embeddings: Redes neuronales recurrentes (RNN) que pueden aprender representaciones con secuencias temporales.
   * Transformer Embeddings: La arquitectura transformer, utilizada en modelos como GPT, puede generar embeddings para secuencias temporales.

La elección del embedding depende del tipo de datos y del problema específico que se aborde. Otro punto interesante es que se pueden combinar.

#### 2.2.2.1 Word Embedding

Se puede definir como una técnica de procesamiento de lenguaje natural que convierte lenguaje humano a vectores matemáticos. Se hace un mapeo de palabras a vectores de números reales, lo que significa que cada palabra está en el vector.

Una vez se tiene la matriz que mapea palabras a vector numérico, se puede realizar acciones aritméticas sobre dichos vectores. Así se puede tener similitudes semánticas (sinónimos) de palabras, frases y aun documentos enteros. Así se puede usar información para determinar programáticamente de que texto se trata.

Matemáticamente, determinar la semántica similar entre dos palabras se reduce a calcular el coseno similar entre el vector correspondiente, o calcular el coseno de el angulo entre los vectores (Vasiliev, 2020))

Para hacer un análisis de cómo se trabaja con vectores de texto se utiliza Spacy que es una biblioteca de procesamiento de lenguaje natural de código abierto diseñada para realizar tareas de procesamiento de texto de manera más eficiente, que ofrece herramientas para realizar actividades como tokenización, lematización, etiquetado gramatical, reconocimiento de entidades nombradas y análisis de dependencias sintácticas.

**Trabajando con vectores de Palabra**

Los vectores de palabras son números reales que permiten que las maquinas entiendan el lenguaje natural. El cálculo de las similitudes de la semántica de diferentes textos, permite por ejemplo clasificar textos según el tópico a cubrir.

Lo primero es un vistazo conceptual en los vectores de palabra, así como una idea de cómo matemáticamente calcula las similitudes semánticas entre palabras representadas en la forma de un vector. Sé mapean palabras a vectores de números reales que reflejan similitudes semánticas de las palabras. Se puede imaginar un espacio de vector de palabras como una nube en el cual el vector de palabras con similitudes está localizado cercanamente. Por ejemplo, un vector representando la palabra “Salvador” debería estar cerca a el vector de la palabra “Mesías”. Para generar estos vectores se debe estar disponible a codificar el significado de estas palabras. Hay un enfoque para codificar el significado

**Definir significado con coordenadas.**

Una manera para generar un vector de palabras significativas es o asignar un objeto o categoría desde el mundo real a cada coordenada de un vector de palabras. Por ejemplo, para las siguientes palabras: Roma, Italia Atenas y Grecia. El vector de palabras debería matemáticamente reflejar de hecho que roma es capital de Italia y Atenas no tiene igual relación con Italia. También que Atenas y roma son ciudades capitales y además que Grecia y Italia son Países.

La Tabla siguiente muestra la matriz de este vector.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | Pais | Capital | Grecia | Italia |
| Italia | 1 | 0 | 0 | 1 |
| Roma | 0 | 1 | 0 | 1 |
| Grecia | 1 | 0 | 1 | 0 |
| Atenas | 0 | 1 | 1 | 0 |

Tabla 1 Simplificado Espacio vectorial de palabras

Así se distribuye el significado de cada palabra entre sus coordenadas en un espacio de cuatro dimensiones, representando las categorías “País”, “Capital”, “Grecia”, “Italia”. En este simple ejemplo una coordenada puede tomar un valor de cero o uno indicando si la palabra pertenece o no a la categoría.

Una vez se tiene el vector de números que corresponden al significado de las palabras, se puede usar el vector aritmético para entender mejor el significado de una palabra. Para descubrir de cual país es capital la ciudad de Atenas se puede usar la siguiente ecuación, donde cada valor representa su vector correspondiente y X es un vector desconocido

*Italia – Roma = X – Atenas*

Esta ecuación expresa una analogía en cual X representa el vector de palabra que tiene la misma relación para Atenas como Italia la tiene para Roma.

Para resolver para X, Se puede reescribir la ecuación como la siguiente:

*X = Italia – Roma + Atenas*

Primero se resta el vector Roma del vector Italia para restar el correspondiente elemento del vector. Así a suma de los vectores es la siguiente

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Operación |  | País | Capital | Grecia | Italia |
|  | Italia | 1 | 0 | 0 | 1 |
| - | Roma | 0 | 1 | 0 | 1 |
| + | Atenas | 0 | 1 | 1 | 0 |
|  | Grecia | 1 | 0 | 1 | 0 |

Tabla 2 Operación matemática vectorial en un espacio vectorial de palabras

Por restar el vector de la palabra Roma de del vector de la palabra Italia y sumar el vector de la palabra Atenas, se tuvo el resultado que es igual al vector Grecia.

**Usando dimensiones para representar significados**

Con el ejemplo anterior se tomaron cuatro categorías, realmente en el mundo real de los vectores de palabras pueden llegar a ser decenas de miles. El vector de palabras de estas proporciones se vuelvo impráctico para muchas aplicaciones, porque esto requiere una matriz inmensa de Word-embedding. Por ejemplo, para trabajar unas 10.000 categorías y 1.000.000 de entidades a codificar, se necesitaría un 10.000 x 1.000.000 matriz de embedding, haciendo que las operaciones de tiempo de consumo demasiado largo. Así para lograr reducir el tamaño la matriz embedding seria reducir el número de categorías que se involucran en el espacio del vector.

En vez de utilizar las coordenadas que representan todas las categorías, se usa la distancia entre vectores para cuantificar y categorizar semánticamente las similitudes. Las dimensiones individuales no tienen el significado en vez de esto tienen las representaciones del espacio en el vector y las distancias entre los vectores indican las similitudes a el correspondiente significado de las palabras.

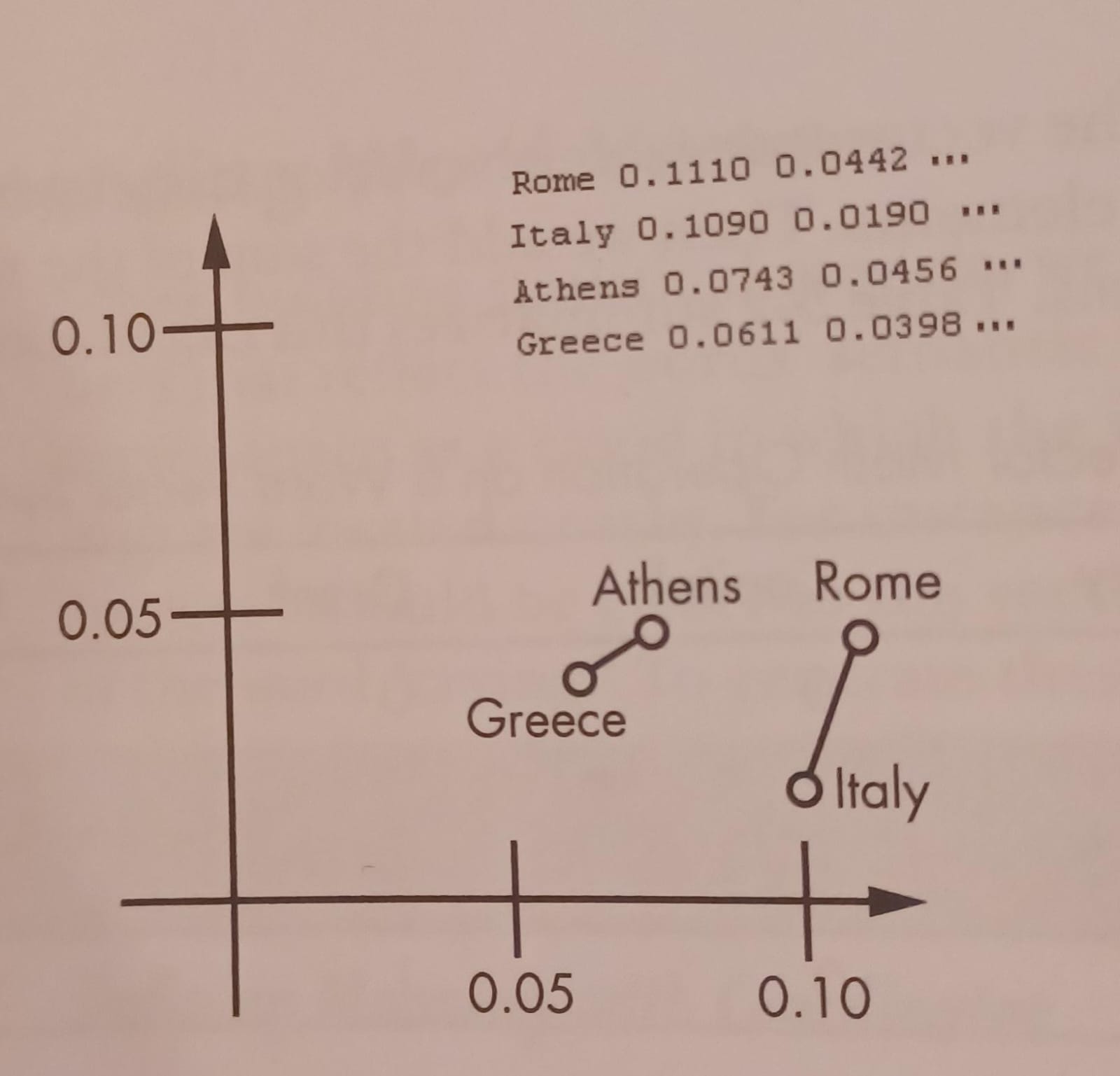


Figura 6 Un fragmento de una proyección de 2D

Una detalle interesante es que las lineas de Grecia-Athenas y Italia-Roma, respectivamente son paralelas, lo que demuestra es que los vectores en el diagrama pais-capital tienen una relacion.

**El metodo de Similitud (similitary)**

En la lbiblioteca de Spacy cada objeto del container tiene un metodo de similitud que permite que se calcule la semantica de similitud entre dos container de cualquier tipo de comparacion entre dos vectores de palabra. Para calclar la similitud de spans y documentos, lo cual no tiene sus propios vectores de palabras spaCy promedia los vectores de palabras de los tokens que ellos contienen.

Se puede calcular la similitud de semantica de dos objetos de container aun si los dos objetos son diferentes. Por ejemplo se puede comparar un objeto token con un objeto span (fragmento de un documento) , un objeto san con un documento y asi sucesivamente.

El siguiente ejemplo calcula dos objetos , una para el documento completo y otro para el fragmento especifivo y luego calcula la similitud entre ellos.

*>>> doc =nlp(‘Yo quiero una manzana verde”)*

*>>> doc.similarity(doc[2:5])*

*0.7305813588233471*

*El codigo anterior es pythhon usando la biblioteca Spacy y la expresión `doc[2:5]` se refiere a una "rebanada" (slice) de tokens en un objeto `Doc` en spaCy. Desglosemos lo que significa:*

*- `doc`: Es un objeto `Doc` que representa un documento procesado por spaCy, que contiene información lingüística sobre el texto.*

*- `[2:5]`: Es una notación de "rebanada" (slice) en Python. En este contexto, selecciona un subconjunto de tokens del objeto `Doc`. En spaCy, la indexación de tokens comienza desde 0. Entonces, `doc[2:5]` selecciona los tokens desde el tercer token hasta el quinto token (sin incluir el quinto).*

*Por ejemplo, si `doc` contiene la frase "Esto es un evento de la antigüedad", entonces `doc[2:5]` seleccionaría los tokens "un", "evento", "de".*

*La notación de rebanada en Python es `[inicio:final]`, y selecciona los elementos desde `inicio` hasta `final-1`. En este caso, selecciona los tokens desde el tercer token (índice 2) hasta el quinto token (índice 4) en el objeto `Doc`.*

Algo interesante es que cuando se realiza la similitud comparado objetos consigo mismo el resultado es un 1.

El metodo de similitud puede reconocer palabras que pertenecen a la misma o similar categoria y que frecuentemente aparecen relacionando contextos, mostrando un alto nivel de similitud escogiendo plabras claves para calculo de similitudes. El metodo de similitud calcula similitudes semanticas, pero el resultado es más útil para el cálculo escogiendo las palabras claves para comparar.

Se puede considerar clasificar el texto en una variedad de diferentes maneras dependiendo del conjunto de categorías que se van a usar. Por ejemplo, Si estamos buscando el texto acerca de las plantas más altas del planeta, la frase “Arboles altos” y “en el mundo” serán las claves. Comprando esta frase con “Plantas Altas” y “sobre el planeta” deberían tener una similitud altísima semánticamente. Se pueden extraer trozos sustantivos usando una propiedad *doc.noun.chunk (propiedad de la biblioteca spaCy que extrae grupos de sustantivos de un documento*) y así chequear la similitud de trozos sustantivos y la búsqueda de frases usando este método.

Si buscamos lugares en el mundo. “Bogotá” será la palabra clave. Realmente no sabemos de antemano cual nombre geopolíticamente puede ocurrir en el texto. Puede ser Bogotá o Amazonas. Pero sea lo que sea, esto debería ser semánticamente similar a la palabra “geografía”. La cual se puede comparar con otros sustantivos del texto. Si se puede determinar que hay un alto grado de similitud, se puede asumir que el nombre de la entidad en cuestión representa el nombre geopolítico.

Se pueden generar los números para poner en vector usando un algoritmo de aprendizaje automático. El Machine Learning, un subcampo de la inteligencia artificial, crea sistemas de computadores que pueden automáticamente aprender de datos que no han sido explícitamente programados. Los algoritmos pueden hacer predicciones acerca de nuevos datos, aprender a reconocer imágenes y discursos, clasificar fotos y documentos de texto, automatizar controles y ayudar en desarrollo de juegos.

El aprendizaje automático (Machine Learning) permite que las computadoras realicen tareas que serían difíciles, sino en muchos casos imposible de hacerlo.

Por ejemplo, para realizar un programa normal de jugar el algoritmo ajedrez, el algoritmo debe contener condiciones de si entonces (if...else) que necesitan ser definidas. Así se desarrolle exitosamente, el programa va a tener puntos débiles en su lógica que pueden tomar ventaja en el mismo juego antes que se hagan correcciones en el código del programa.

En contraste las aplicaciones construidas sobre machine Learning no confían en la predeterminada, pero si en la capacidad de aprender de experiencias del pasado. La aplicación mira las posiciones jugadas y recuerda de juegos pasados y hace el movimiento a la mejor posición. Almacena las experiencias pasadas en un modelo estadístico.

**Modelo Estadísticos**

En Spacy, además de generar vectores de palabras, permite lograr tres tareas: Análisis de dependencia semántica (determinar la relación entre las palabras en la oración), parte del etiquetado del discurso (identificar sustantivos, verbos y otros partes de la oración y reconocimiento de la entidad nombrada (organizar los sustantivos apropiados dentro de categorías como gente, organizaciones y localizaciones.

El ciclo típico del Machine Learning tiene tres pasos:

**Entrenamiento Modelo**

La primera fase es alimentar el algoritmo con una gran cantidad de datos. Para que sea confiable se debe proveer de una suficiente cantidad de datos en la entrada, al pensar en NPL se puede tener plataformas como Wikipedia y Google News que contiene suficiente texto para alimentar virtualmente cualquier algoritmo de aprendizaje automático (machine Learning). Para modelos específicos se debe buscar sitios que puedan ser para el caso de estudio.

Figura 7 Generando un modelo estadístico con el algoritmo de aprendizaje automático

La figura anterior proporciona una descripción de alto nivel de la etapa de entrenamiento del modelo.

El modelo procesa grandes volúmenes de datos de texto para entender cuales palabras comparten característica, entonces este crea vectores de palabra para aquellas palabras que reflejan características que comparten

El espacio del vector de palabras no solo tiene el componente del modelo estadístico. La estructura actual es típicamente más complicada, provee una manera de extraer las características lingüísticas para cada palabra dependiendo del contexto en donde aparece.

#### 2.2.2.2 Pruebas

Una vez que el modelo esta entrenado, se realizan pruebas para saber que tan bien esta la ejecución. Para hacer pruebas se alimenta el texto que no ha sido alimentado antes y así comprobar si puede identificar con éxito las similitudes semánticas y otras características aprendidas durante el entrenamiento.

#### 2.2.2.3 Haciendo Predicciones

Con las pruebas funcionando se pasa a usar el modelo para hacer predicciones en su aplicación de PNL. Por ejemplo, puede usarlo para predecir una estructura de árbol de dependencia sobre el texto que ingresa, como se muestra en la Figura 1-2. Una estructura de árbol de dependencia representa las relaciones entre las palabras en una oración.

Arbol d

Figura 8 Ejemplo Predicción una estructura de árbol de dependencia usando un modelo estadístico

### 2.2.3 Google BERT (Bidirectional Encoder Representations from Transformers) to

Es un modelo de procesamiento de lenguaje natural desarrollado por Google. Utiliza la arquitectura de Transformers para entender el contexto de las palabras en una oración, teniendo en cuenta tanto palabras anteriores como siguientes. Esto mejora la capacidad del modelo para comprender el significado de una palabra en función de su contexto, lo que resulta en una mejor comprensión del lenguaje natural y una mejora de la calidad de las respuestas generadas por los modelos de procesamiento de lenguaje natural.

(Ravichandiran, January 2021) compartir sus proyectos de software de manera centralizada.

### 2.2.4. Herramientas Colaborativas

#### 2.2.4.1 GitHub

Usar una plataforma que permita la colaboración en este proyecto puede ser la base para que analistas de datos con interés en el procesamiento natural del lenguaje, sea experimentado que puedan aportar en ideas o en el propio desarrollo pueden ser un aspecto importante para tener éxito en el proyecto que se está desarrollando.

GitHub es una plataforma de desarrollo colaborativo que utiliza el sistema de control de versiones Git, Permite a los desarrolladores trabajar juntos en proyectos de software, facilitando el seguimiento de cambios en el código fuente, la colaboración entre equipos y la gestión eficiente de proyectos. Los usuarios pueden alojar sus repositorios de código en GitHub, lo que significa que pueden almacenar.

Algunos aspectos útiles para desarrollar el proyecto:

**Control de versiones:** Con lo que facilita un contexto de trabajo colaborativo. Lleva el control de versiones muy importante en el mundo donde una o varias personas están desarrollando un proyecto,

**Repositorios**: Es el espacio donde se almacena todo el código fuente, archivos de configuración, documentos otros recursos relacionados con un proyecto. Puede ser público (accesible para todos) o privado (accesible solo para personas autorizadas).

**Colaboración de desarrolladores,**

Seguimiento a problemas(bugs) Los usuarios pueden informar bugs, solicitar nuevas características o discutir ideas a través del seguimiento de problemas GitHub.

**Despliegue:** Utiliza un despliegue en sitios de web y aplicaciones a través de GitHub Pages, una función que permite alojar sitios web directamente desde los repositorios de GitHub

GitHub crea un ambiente que permite almacenar código en un servidor remoto, dando la habilidad de compartir el código con otras personas y hacer más fácil que una persona agregue, modifique o borre código del mismo archivo y proyecto, mientras que guarda una fuente de verdad para este archivo. (Sarah Guthals, 2019)

#### *2.2.4.2* Google *Colab*

Herramientas colaborativa que permite programar y ejecutar Python con acceso a GPUs y TPU que puede acelerar el entrenamiento de modelos de aprendizaje automático  de uso gratuito que permite compartir contenido fácilmente para brindar acceso a la mayor cantidad posible de grupos en todo el mundo, Colab prioriza a los usuarios que programan activamente en un cuaderno, también restringe las acciones que impactan negativamente a otros o que están asociadas con eludir nuestras políticas antiabuso.

Todos los cuadernos de Colab se almacenan en Google Drive o puedes cargar desde GitHub. Los cuadernos de Colab se pueden compartir igual que los archivos de Documentos de Google.

Además se puede aprovechar toda la potencia de las bibliotecas más populares de Python para analizar y visualizar datos. La celda de código de abajo utiliza NumPy para generar datos aleatorios y Matplotlib para visualizarlos. En el trabajo en IA puede ser necesario acceder a datos con SQL.

### 2.2.5. Python

Python es una herramienta ampliamente utilizada en el análisis de texto ya que tiene varias bibliotecas y frameworks especializados en NLP y análisis de texto. Entre ellos se destacan spaCy, NLTK, Textblob, Gemsis y scikti-leand que proporcionan herramientas poderosas para tareas como tokenizacion, análisis sintáctico, análisis de sentimiento, modelado de temas, entre otros.

Su integralidad con otras tecnologías y herramientas utilizadas en ciencia de datos y análisis, como jupyter Notebooks, pandas para manipulación de datos, matplotlib y seaborn para visualización y tensrflow o pytorch para aprendizaje profundo la hace una herramienta ideal para trabajar con el proyecto.

La academia, la industria y hasta los gobiernos han adoptado ampliamente a Python como una herramienta que facilita la colaboración y el intercambio de código y prácticas de análisis de texto.

#### 2.2.5.1 Análisis de Texto

En Python el texto se puede manejar en forma de cadena (string) (Python string common operations, n.d.), y tipo de secuencia de texto (str), que son objetos de la clase str (Python - Built-in types, n.d.). Python 3 maneja texto como Unicode de forma predeterminada, mientras que Python 2 trata el texto como bytes. Esto es crucial para trabajar con texto en varios idiomas, ya que Unicode es un estándar que incluye la mayoría de los caracteres utilizados en diferentes escrituras del mundo

Unicode es simplemente un lenguaje de codificación o una forma en que manejamos el texto. Por ejemplo, el valor Unicode para la letra Z es U+005A. Hay muchos tipos de codificación e históricamente en Python, se esperaba que los desarrolladores manejaran diferentes codificaciones por su cuenta, con todas las acciones de bajo nivel ocurriendo en bytes. De hecho, el cambio en la forma en que Python maneja Unicode ha generado muchas discusiones (Bramlett, 2016), críticas (Ronacher, 2014)y elogios (Coghlan, 2014) dentro de la comunidad.

Hablamos de Google usando TensorFlow y Apple usando Scikit-learn, por ejemplo. El código fuente abierto está alcanzando los mismos estándares y eficiencia que el código industrial; una de las bibliotecas en las que nos centraremos a lo largo de este libro, spaCy, es un ejemplo de esto. La recopilación de datos también se realiza en gran medida con Python, utilizando bibliotecas como tweepy (Twitter), urllib (acceso a páginas web) y beautiful sopa (extracción de HTML de páginas web). Que más personas usen un determinado ecosistema significa que crecerá (la publicación del blog Stack Overflow hace un buen artículo sobre esto [6]), y esto significa que tanto los investigadores como la industria lo están usando cada vez más, lo que significa que es un buen momento para dar el salto. ¡súbete al carro!

Aparte del soporte externo que recibe Python de la amplia variedad de bibliotecas (y en particular, de las bibliotecas de NLP), existen otras razones por las que Python es un lenguaje atractivo de usar. Uno de ellos es el uso predominante de Python como lenguaje de programación. Un lenguaje de secuencias de comandos es aquel en el que se admite la capacidad de ejecutar secuencias de comandos; programas escritos para un entorno de ejecución que normalmente automatiza tareas. Por ejemplo, si escribes unas cuantas líneas de código para responder rápidamente a los deseos de cumpleaños de Facebook, y esto se hace todos los años, es un ejemplo de script. No existe una regla estricta para lo que se llama lenguaje de secuencias de comandos, sino más bien una forma en que discutimos coloquialmente los lenguajes de programación.

Aparte del soporte externo que recibe Python de la amplia variedad de bibliotecas (y en particular, de las bibliotecas de NLP), existen otras razones por las que Python es un lenguaje atractivo de usar. Uno de ellos es el uso predominante de Python como lenguaje de programación. Un lenguaje de secuencias de comandos es aquel en el que se admite la capacidad de ejecutar secuencias de comandos; programas escritos para un entorno de ejecución que normalmente automatiza tareas. Por ejemplo, si escribes unas cuantas líneas de código para responder rápidamente a los deseos de cumpleaños de Facebook, y esto se hace todos los años, es un ejemplo de script. No existe una regla estricta para lo que se llama lenguaje de programación, pero es más bien una forma en que discutimos coloquialmente los lenguajes de programación.

Python es un lenguaje de secuencias de comandos muy útil debido a la rapidez con la que se puede codificar una secuencia de comandos para manipular archivos de texto: es fácilmente legible, lo suficientemente rápido para tamaños de archivos que no son masivos y es un lenguaje interpretado, lo que significa que no se necesita compilar el código antes de ejecutarlo. Se escribe dinámicamente, lo que significa que no se necesita definir tipos de datos mientras escribimos código. Pero más que las razones técnicas de por qué Python es superior, se está más interesado ​​en Python por su facilidad de uso. Es flexible, legible y con un alto nivel de abstracción, lo que permite se sea más productivo. Se puede centrar más en el problema que en los tecnicismos de programación y errores de código. Esto no quiere decir que no se tienen errores de código al codificar en Python; sólo que tienden a tener más solución y proporcionar más información que solo, por ejemplo: FALLO DE SEGMENTACIÓN.

#### 2.2.5.2 Bibliotecas Python

Para el NLP procesamiento de lenguaje natural hay varias bibliotecas poderosas y populares como son:

**NLTK**: Es una biblioteca extensa para trabajar con datos de texto. Proporciona herramientas de tokenizacion, stemming, lematización, análisis sintético y más. (NLTK, Consultada en 2023)

**spaCy**: Es una biblioteca abierta, diseñada para ser rápida y eficiente. Proporciona modelo preentrenados. Tiene muy buen rendimiento en grandes cantidades de texto.

**TextBlob**: Es una biblioteca fácil de usar para el procesamiento de texto basado en NLTK. Ofrece una API para tareas sencillas, como análisis de sentimiento, extracción de frases clave y clasificación de texto. (Textblob: Simplified Text Processing, Consultado en 2023)

**Gensim**: Esta biblioteca se centra en el modelado de temas y similitud de documentos. Es útil para la construcción y entrenamiento de modelo de vectores de palabras (Word embedddings) y modelos de tópicos. (GENSIM topic modeling for humans, Consultada en Nov 22 2023)

**Transformer**: Proporciona implementaciones de modelos de lenguaje preentrenados de vanguardia, como BERT, GPT-2. (Tranformers Huggin Face, Consultado en nov 22 2023)

#### 2.2.5.3 spaCy

Con el estudio de esta biblioteca se puede revisar los conceptos en la cadena de procesamiento o pipeline que se dan en los pasos básicos en el NLP. Las operaciones incluyen Tokenización, lematización, etiquetado, dependencia sintáctica y reconocimiento de nombres de entidades. (Vasiliev, 2020)

La Figura siguiente proporciona una descripción simplificada de este proceso:

Objeto Salida

Texto Entrada

Reconocimiento entidades

Lematización

Tokenización

Analizando

Etiquetas

Figura 9 Vistazo General de los pasos de NLP

El proceso casi siempre incluye estos pasos Tokenización, lematización, etiquetado de parte del párrafo, análisis y reconocer los nombres de las entidades.

##### 2.2.5.3.1 Tokenización

El primer paso dentro del análisis del texto es lo denominado como tokens que es pasar a números las palabras. Utilizando Python y spaCy se puede revisar el siguiente código para entenderlo.



Figura 10 Ejemplo Tokenización

EL ejemplo se corre en la plataforma de Google Colab y el primer paso es cargar la librería spaCy, en el paso 2 cargar una paquete objeto lenguaje *es\_core\_news\_sm* que contiene el vocabulario del idioma y otros datos del modelo estadístico. Se llama al objeto de lenguaje nlp y a continuación, se aplica el objeto recién creado a una oración de muestra, creando una instancia de objeto doc. Un objeto doc es un contenedor para una secuencia de objetos Token, spaCy lo genera implícitamente en función del texto proporcionado. En este punto, con pocas líneas de código, spaCy ha generado la estructura gramatical para la oración de muestra. Cómo se utiliza depende de la necesidad del proyecto. En este ejemplo muy simple, simplemente imprima el contenido del texto de cada ficha de la oración de muestra.

El script genera los tokens de la oración de muestra como una lista:

['Yo', 'soy', 'volando', 'a', 'Frisco']

El contenido del texto (el grupo de caracteres que componen el token, como las letras "s", “o” y "y" en el token "soy") es sólo una de las muchas propiedades de un objeto Token. También se puede extraer varias características lingüísticas asignadas a un token.

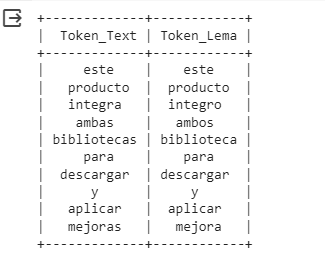
##### 2.2.5.3.2 Lematización

Un lema es la forma base de un token. Es la forma en que aparecería el token si estuviera incluido en un diccionario. Por ejemplo, el lema de la ficha "volando" es "volar". La lematización es el proceso de reducir las formas de las palabras a su lema. El siguiente script proporciona un ejemplo sencillo de cómo realizar la lematización con espacios:



Figura 11 Ejemplo de Lematización

Las primeras líneas son las mismas que las del código de Tokenización creando un objeto doc a través del cual puede acceder a la estructura gramatical de la oración. En la gramática, la estructura de una oración es la disposición de palabras individuales, así como de frases y cláusulas en una oración. El significado gramatical de una oración depende de esta organización estructural. Una vez que tenga un objeto Doc que contenga los tokens de su oración de ejemplo, itere sobre esos tokens en un bucle y luego imprima el contenido de texto de un token junto con su lema correspondiente. Este script produce el siguiente resultado



La columna de la izquierda contiene las fichas y la columna de la derecha contiene sus lemas.

**Aplicar la lematización para el reconocimiento de significado**

La lematización es un paso importante en la tarea de reconocimiento de significado. Para ver cómo, volvamos a la frase de ejemplo de la sección anterior:

Estoy volando a Frisco.

Supongamos que esta oración se envió a una aplicación de PNL que interactúa con un sistema en línea que proporciona una API para reservar boletos para viajes. La aplicación procesa la solicitud de un cliente, extrae de ella la información necesaria y luego la pasa a la API subyacente. Este diseño podría parecerse al que se muestra en la Figura 2-2.

El proceso de procesamiento de textos 19

Estoy volando a Frisco

solicitud de PIN

**Forma de viaje: volar**

**Destino: San Francisco**

**API de reserva de billetes en línea**

**Figura 2-2: Uso de la lematización en el proceso de extracción de la información necesaria de la solicitud de un cliente**

**La aplicación PNL intenta obtener la siguiente información a partir de la solicitud de un cliente: una forma de viaje (avión, tren, autobús, etc.) y un destino. La aplicación debe determinar primero si el cliente desea un billete de avión, de tren o de autobús. Para determinar esto, la aplicación busca una palabra que coincida con una de las palabras clave en la lista predefinida. Una forma sencilla de simplificar la búsqueda de estas palabras clave es convertir primero todas las palabras de una oración que se está procesando en sus lemas. En ese caso, la lista predefinida de palabras clave será mucho más corta y clara. Por ejemplo, no será necesario incluir todas las formas de la palabra volar (como "volar", "volar", "volar" y "volar") para que sirva como indicador de que el cliente quiere un boleto aéreo. , reduciendo todas las variantes posibles a la forma básica de la palabra, es decir, "volar".**

**La lematización también resulta útil cuando la aplicación intenta determinar un destino a partir de una solicitud enviada. Hay muchos apodos para las ciudades del mundo. Pero el sistema que reserva los billetes requiere nombres oficiales. Por supuesto, el Tokenizer predeterminado que realiza la lematización no sabrá la diferencia entre apodos y nombres oficiales de ciudades, países, etc. Para resolver este problema, puede agregar reglas de casos especiales a una instancia de Tokenizer existente.**

**El siguiente script ilustra cómo podría implementar la lematización para el ejemplo de las ciudades de destino. Imprime los lemas de las palabras que componen la oración.**

**importar espacio**

**desde spacy.symbols importa ORTH, LEMMA**

**pnl spacy.load('es')**

**doc = nlp(u'Estoy volando a Frisco')**

**print([w.texto para w en doc])**

**• caso\_especial = [(ORTO: u'Frisco', LEMA: u'San Francisco'}]**

**nlp.tokenizer.add\_special\_case(u'Frisco', special\_case) print([w.lemma para w en nlp(u'Estoy volando a Frisco')])**

**Se define un caso especial para la palabra Frisco reemplazando su lema predeterminado por San Francisco. Luego agrega este caso especial a la instancia de Tokenizer. Una vez agregada, la instancia de Tokenizer usará este caso especial cada vez que se le solicite el lema de Frisco. Para asegurarse de que todo funcione como se esperaba, imprima los lemas de las palabras de la oración. El script genera el siguiente resultado:**

**['yo', 'soy', 'volando', 'a', 'Frisco']**

**['-PRON-', 'estar', 'volar', 'a', 'San Francisco']**

**El resultado enumera los lemas de todas las palabras que aparecen en la oración con la excepción de Frisco, para el cual enumera San Francisco.**

**Etiquetado de parte del discurso**

**Una etiqueta de parte del discurso le indica la parte del discurso (sustantivo, verbo, etc.) de una palabra determinada en una oración determinada. (Recuerde del Capítulo 1 que una palabra puede actuar como más de una parte del discurso dependiendo del contexto en el que aparece).**

**En spaCy, las etiquetas de parte del discurso pueden incluir información detallada sobre un token. En el caso de los verbos, podrían indicarle las siguientes características: tiempo (pasado, presente o futuro), aspecto (simple, progresivo o perfecto), persona (1.ª, 2.ª o 3.ª) y número (singular o plural). ).**

**Extraer estas etiquetas de partes del discurso verbal puede ayudar a identificar la intención de un usuario cuando la tokenización y la lematización por sí solas no son suficientes. Por ejemplo, el script de lematización para la aplicación de reserva de boletos en la sección anterior no decidirá cómo la aplicación de PNL elige palabras en una oración para redactar una solicitud a la API subyacente. En una situación real, hacerlo podría resultar bastante complicado. Por ejemplo, la solicitud de un cliente puede constar de más de una frase:**

**He volado a Los Ángeles. Ahora estoy volando a Frisco.**

**Para estas oraciones, los resultados de la lematización serían los siguientes:**

**['-PRON-', 'tener', 'volar', 'a', 'LA', '', 'ahora', '-PRON-', 'ser', 'volar', 'a', 'San Francisco','.]**

**En este caso, realizar la lematización por sí sola no es suficiente; la aplicación podría**

**considere los lemas "fly" y "LA" de la primera oración como palabras clave. indicando que el cliente tiene la intención de volar a Los Ángeles cuando en realidad el cliente tiene la intención de volar a San Francisco. Parte del problema es que la lematización cambia los verbos a sus formas infinitivas, lo que dificulta saber el papel que desempeñan en una oración. Aquí es donde entran en juego las etiquetas de parte del discurso. En inglés, el núcleo**

**Las partes del discurso incluyen sustantivo, pronombre, determinante, adjetivo, verbo, adverbio, preposición, conjunción e interjección. (Consulte el manual lingüístico en el apéndice para obtener más información sobre estas partes del discurso). En spaCy, estas mismas categorías, más algunas adicionales para símbolos, puntuación**

**El proceso de procesamiento de textos 21**

**Las marcas y otras se denominan partes del discurso específicas del curso y están disponibles como un conjunto fijo de etiquetas a través de los atributos Token.pos (int) y Token.pos (unicode).**

**Además, spaCy ofrece etiquetas de partes del discurso detalladas que brindan información más detallada sobre un token, cubriendo características morfológicas, como tiempos verbales y tipos de pronombres. Naturalmente, la lista de partes del discurso detalladas contiene muchas más etiquetas que la lista detallada. Las etiquetas de parte del discurso detalladas están disponibles como Token. etiqueta (int) y token. atributos de etiqueta (Unicode).**

**La Tabla 2-1 enumera algunas de las etiquetas de partes del discurso comunes utilizadas en spaCy para modelos en inglés.**

**Tabla 2-1: Algunas etiquetas comunes de partes del discurso spaCy**

**TAG (POS de grano fino (de grano grueso)**

**Parte del discurso)**

**NN**

**ENN**

**PPR**

**PRP**

**V.B.**

**VBD**

**JBV**

**Parte del discurso)**

**Morfología**

**Descripción**

**SUSTANTIVO**

**cantar números**

**Sustantivo, singular**

**SUSTANTIVO**

**Número=plur**

**sustantivo, plural**

**PRON**

**PronType=prs**

**pronombre personal**

**PRON**

**PronType=prs**

**Pos-sí**

**Pronombre, posesivo**

**VERBO**

**Forma verbal-inf**

**Verbo, forma base**

**VERBO**

**Forma verbal = fin tiempo pasado**

**Verbo, tiempo pasado**

**VERBO**

**Forma verbal = parte**

**tiempo-pres**

**Verbo, gerundio o participio presente**

**beneficios según objetivos**

**2**

**adj.**

**Prog de aspecto**

**Posgrado**

**Adjetivo**

**Puede encontrar la lista completa de etiquetas detalladas de partes del discurso utilizadas en spaCy en la sección "Etiquetado de partes del discurso" en el manual de Especificaciones de anotación en https://spacy.io/api/annotation#pos -etiquetado.**

**El tiempo y el aspecto son quizás las propiedades más interesantes de los verbos para aplicaciones de PNL. Juntos, indican la referencia de un verbo a una posición en el tiempo. Por ejemplo, utilizamos la forma de aspecto progresivo en tiempo presente de un verbo para describir lo que está sucediendo ahora o lo que sucederá en el futuro cercano. Para formar el verbo de aspecto progresivo en tiempo presente, se agrega la forma en tiempo presente del verbo "to be" antes de un verbo -ing. Por ejemplo, en la oración "Estoy investigando", agrega "soy", la forma del verbo "ser" en primera persona, tiempo presente, antes del verbo -ing "mirar". En este ejemplo, "am" indica el tiempo presente y "mirando" apunta al aspecto progresivo.**

**Uso de etiquetas de parte del discurso para encontrar verbos relevantes**

**La aplicación de reserva de boletos podría usar las etiquetas detalladas de partes del discurso disponibles en spaCy para filtrar los verbos en el discurso, eligiendo solo aquellos que podrían ser clave para determinar la intención del cliente.**

**Antes de pasar al código de este proceso, intentemos descubrir qué tipo de expresiones podría utilizar un cliente para expresar su intención de reservar un billete de avión a, por ejemplo, Los Ángeles. Podríamos empezar viendo algunas oraciones que contengan la siguiente combinación de lemas: "fly", "to" y "LA". Aquí hay algunas opciones simples:**

**Volé a Los Ángeles.**

**He volado a Los Ángeles.**

**Necesito volar a Los Ángeles.**

**Estoy volando a Los Ángeles.**

**Volaré a Los Ángeles.**

**Tenga en cuenta que, aunque todas estas frases incluirían la combinación "volar a Los Ángeles" si se redujeran a lemas, sólo algunas de ellas implican la intención del cliente de reservar un billete de avión a Los Ángeles. Los dos primeros definitivamente no son adecuados. Un análisis rápido revela que las formas pasadas y pasadas perfectas del verbo "volar" (los tiempos utilizados en las dos primeras oraciones) no implican la intención que estamos buscando. Sólo son adecuadas las formas infinitivo y presente progresivo. El siguiente guión ilustra cómo encontrar esas formas en el discurso de muestra:**

**importar espacio**

**pnl spacy.load('es')**

**docnip(u'He volado a Los Ángeles. Ahora estoy volando a Frisco.') print([w.text for win doc if Ow.tag == 'VBG' or w.tag== 'VB'])**

**La propiedad de etiqueta de un objeto Token contiene la parte detallada**

**atributo de habla asignado a ese objeto. Utiliza un bucle realizado sobre**

**los tokens que componen el discurso para comprobar si el detalle**

**La etiqueta de parte del discurso asignada a un token es VB (un verbo en la forma base o infinitiva) o VBG (un verbo en la forma presente progresiva). En el discurso de muestra, sólo el verbo "volar" en la segunda oración cumple la condición especificada. Entonces deberías ver el siguiente resultado:**

**['volador"]**

**Por supuesto, las etiquetas detalladas de partes del discurso no solo se asignan a los verbos; también se asignan a las otras partes del discurso en una oración. Por ejemplo, spaCy reconocería LA y Frisco como nombres propios (sustantivos que son nombres de personas, lugares, objetos u organizaciones) y los etiquetaría con PROPN. Si lo desea, puede agregar la siguiente línea de código al script anterior:**

**print({w.text para win doc si w.pas "PROPN'))**

**El taxi disparando peine**

**23**

**Agregar ese código debería generar la siguiente lista:**

**[LA', "Frisco"]**

**Los nombres propios de ambas oraciones del discurso de muestra están en la lista.**

**El contexto es importante**

**Es posible que las etiquetas detalladas de partes del discurso no siempre sean suficientes para determinar el significado de una expresión. Para ello, es posible que aún tengas que confiar en el contexto. Como ejemplo, considere la siguiente expresión: "Estoy volando a Los Ángeles". El etiquetador de parte del discurso asignará la etiqueta VBG al verbo "volar" en este ejemplo, porque está en la forma presente progresiva. Pero debido a que usamos esta forma verbal para describir lo que está sucediendo ahora o lo que sucederá en el futuro cercano, el enunciado podría significar "Ya estoy en el cielo, volando a Los Ángeles". o "Voy a volar a Los Ángeles". Cuando se envía a la aplicación PNL de reserva de boletos, la aplicación debe interpretar solo una de estas oraciones como "Necesito un boleto aéreo a Los Ángeles". De manera similar, considere el siguiente discurso: "Voy a volar a Los Ángeles. Por la tarde tengo que regresar a Frisco". Lo más probable es que esto implique que el hablante quiere un billete de avión de Los Ángeles a Frisco para un vuelo nocturno. Encontrará más ejemplos sobre cómo reconocer el significado según el contexto en "Usar el contexto para mejorar el chatbot de reserva de entradas en la página 91.**

**Relaciones sintácticas**

**Ahora combinemos los nombres propios con el verbo que el etiquetador de parte del discurso seleccionó anteriormente. Recuerde que la lista de verbos que podría utilizar para identificar la intención del discurso contiene sólo el verbo "volar" en la segunda oración. ¿Cómo se puede encontrar el par verbo/nombre propio que mejor describa la intención detrás del discurso? Un ser humano obviamente componería los pares verbo/nombre propio a partir de palabras que se encuentran en la misma oración. Debido a que el verbo "volar" en la primera oración no cumple con la condición especificada (recuerde que solo las formas infinitivo y presente progresivo cumplen la condición), podrá componer un par de este tipo solo para la segunda oración: "volar, Frisco."**

**Para manejar estas situaciones mediante programación, spaCy presenta un analizador de dependencia sintáctica que descubre relaciones sintácticas entre tokens individuales en una oración y conecta pares de palabras relacionadas sintácticamente con un solo arco.**

**Al igual que los lemas y las etiquetas de parte del discurso discutidos en las secciones anteriores, las etiquetas de dependencia ryntactic son características lingüísticas que spaCy asigna a los objetos Token que componen un texto contenido en un objeto Doc. Por ejemplo, la etiqueta de dependencia dobj significa "objeto directo". Podríamos ilustrar la relación sintáctica que representa como una flecha, como se muestra en**

**Figura 2-3.**

**HEAD AND CHILD**

**A syntactic dependency label describes the type of syntactic relation between two words in a sentence. In such a pair, one word is the syntactic governor (also called the head or parent) and the other is the dependent (also called the child) spaCy assigns a syntactic dependency label to the pair's dependent. For example, in the pair "need, ticket," extracted from the sentence "I need a plane ficket," the word "ticket" is the child and word "need" is the head, because "need" is the verb in what's called a verb phrase. In this same sentence, "a plane ticket" is a noun phrase: the noun "ticket" is the head, and "a" and "plane" are its children. To learn more, consult "Dependency Grammars vs Phrase Structure Grammars" on page 185.**

**Each word in a sentence has exactly one head. Consequently, a word can be a child only to one head. The opposite is not always the case. The same word can act as a head in none, one, or several pairs. The latter means that the head has several children. This explains why a dependency label is always assigned to the child.**

**Dependency label**

**Head**

**Child**

**need**

**VERB**

**ticket NOUN**

**Coarse-grained. part-of-speech togs**

**Figure 2-3: A graphical representation of a syntactic dependency arc**

**The dobj label is assigned to the word "ticket" because it's the child**

**of the relation. A dependency label is always assigned to the child. In your script, you can determine the head of a relation using the Token.head attribute.**

**You might also want to look at the other head/child relations in the sentence, like the ones shown in Figure 2-4.**

**dobj**

**need**

**plane**

**compound**

**licket**

**Figure 2-4: Head/child relations in an entire sentence**

**The Test Placessing Fipейте 25**

**Como puedes ver, una misma palabra en una oración puede participar en varias relaciones sintácticas. La Tabla 2-2 enumera algunas de las etiquetas de dependencia de inglés más utilizadas.**

**Tabla 2-2: Algunas etiquetas de dependencia comunes**

**etiqueta de dependencia**

**Descripción**

**acompañar**

**Complemento adjetivo**

**amod**

**auxiliar**

**modificador adjetivo**

**Auxiliar**

**compuesto**

**Compuesto**

**dativo**

**Dativo**

**det**

**Determinante**

**dobj**

**Objeto directo**

**nsubj**

**Asunto nominal**

**pobj**

**Objeto de preposición**

**RAÍZ**

**Raíz**

**La etiqueta R00T marca el token cuya cabeza es él mismo. Normalmente, spaCy lo asigna al verbo principal de la oración (el verbo que está en el centro del predicado). Cada oración completa debe tener un verbo con la etiqueta ROOT y un sujeto con la etiqueta nsubj. Los demás elementos son opcionales.**

**beneficios según objetivos**

**La mayoría de los ejemplos de este libro asumirán que el texto enviado es una oración completa y utilizarán la etiqueta ROOT para localizar el verbo principal de la oración. Tenga en cuenta que esto no funcionará para todas las entradas posibles.**

**El siguiente script ilustra cómo acceder a las etiquetas de dependencia sintáctica de los tokens en el discurso del ejemplo de "Etiquetado de partes del discurso" en la página 21:**

**importar espacio**

**pnl spacy.load('es')**

**doc = nlp(u'He volado a Los Ángeles. Ahora estoy volando a Frisco.')**

**para token en doc:**

**imprimir(token.texto, Otoken.pos, token.dep\_)**

**El script genera las etiquetas de parte del discurso generales (consulte la Tabla 2-1) y las etiquetas de dependencia asignadas a los tokens que componen el discurso de muestra:**

**PRON nsub]**

**tener VERBO auxiliar**

**RAÍZ DEL VERBO volado**

**a**

**preparación de ADP**

**PRON SOP preparación Cara PROPE زش**

**PUNTO PUNTO**

**Pero lo que no muestra es cómo se relacionan las maravillas entre sí en la oración z mediante la comúnmente llamada dependencia que se explica al principio de esta sección. Para ver los arcos de dependencia en el discurso de muestra, reemplace el bucle en el script anterior con el siguiente.**

**token lejano en doc**

**imprimir(token.head,text, token.dep, token.text)**

**La propiedad principal de un objeto token se refiere al encabezado sintáctico de este token. Cuando imprimas esta línea, verás cómo las palabras en las oraciones del discurso están conectadas entre sí mediante dependencias sintácticas. Si se presentaran gráficamente, vería un arco para cada línea en la siguiente salida, excepto la relación de 2007. La razón es que la palabra a la que se asigna esta etiqueta es la única palabra en una oración que no tiene encabezado.**

**volado sub) 1 flowe aux tiene ROOT volado flove prep to to pobj LA volado punct volando admod Now Gying sabj Flying aur Flying 1001 volando Flying prep ta to**

**punto volador**

**Mirando la lista anterior de dependencias sintácticas, intentemos descubrir qué etiquetas pintan en los tokens que potencialmente podrían describir mejor la intención del cliente: en otras palabras, necesita encontrar un par que por sí solo describa apropiadamente la intención del cliente.**

**Es posible que le interesen los tokens marcados con las etiquetas ROOT y paty dependency, ya que en este ejemplo son clave para el reconocimiento intenso. Como se indicó anteriormente, la etiqueta marca el verbo principal de se y poj; en este ejemplo, marca la entidad que, en conjunto con las serbummarines, da el significado de todo el enunciado.**

**El siguiente script localiza las palabras asignadas a esas dos dependencias.**

**etiquetas de dencia:**

**importar espacio**

**elp spacy.load('es')**

**docnip(u'He volado a Los Ángeles. Ahora estoy volando a Frisco.')**

**para enviado en doc.sents:**

**print(texto enviado si w.dep\_ROOT' o w.dep 'pobj'))**

**En este guión, trituras el discurso para separar las oraciones con el documento. propiedad sents, que itera sobre las oraciones en el documento. Triturar un texto en oraciones separadas puede ser útil cuando necesita encontrar, por ejemplo, ciertas partes del discurso en cada oración del discurso (analizaremos doc.sents en el siguiente capítulo, donde verá un ejemplo de cómo hacer referencia a los tokens en un documento con índices a nivel de oración). Esto le permite crear una lista de palabras clave potenciales para cada oración basada en etiquetas de dependencia específicas asignadas a los tokens. Las condiciones de filtro utilizadas en este ejemplo se eligen basándose en el examen de los pares sintácticamente relacionados generados por el script anterior. En particular, recoges los tokens con etiquetas de dependencia ROOT y pobj, porque estos tokens forman los pares que te interesan.**

**La salida del script debería verse de la siguiente manera:**

**['flon', 'LA']**

**['volando', 'Frisco"]**

**En ambos pares de oraciones, los sustantivos de salida son los etiquetados como pobj. Puedes usar esto en tu aplicación de reserva de boletos para elegir el sustantivo que mejor combine con el verbo. En este caso, sería "volar", que va con "Frisco".**

**Este es un ejemplo simplificado de extracción de información utilizando etiquetas de dependencia. En los siguientes capítulos, se le brindarán ejemplos más sofisticados de cómo iterar sobre el árbol de dependencia de una oración o incluso de un discurso completo, extrayendo la información necesaria.**

**Prueba esto**

**Ahora que sabe cómo aprovechar la lematización, las etiquetas de parte del discurso y las etiquetas de dependencia sintáctica, puede juntarlas todas para hacer algo útil. Intente combinar los ejemplos de las secciones anteriores en un único guión que identifique correctamente la intención del orador de volar a San Francisco.**

**Su script debería generar el siguiente resultado:**

**['volar, San Francisco)**

**Para lograr esto, comience con el último script de esta sección y mejore la cláusula condicional en el bucle, agregando las condiciones a**

**tenga en cuenta las etiquetas detalladas de partes del discurso, como se explica en "Etiquetado de partes del discurso en la página 21. Luego agregue la funcionalidad de lematización a su secuencia de comandos, como se explica en "Lematización" en la página 18.**

**Reconocimiento de entidad nombrada**

**Una entidad con nombre es un objeto real al que puede hacer referencia mediante un nombre propio. Puede ser una persona, organización, ubicación u otra entidad. Las entidades con nombre son importantes en PNL porque revelan el lugar u organización de la que habla el usuario. El siguiente script busca entidades con nombre en el discurso de muestra utilizado en los ejemplos anteriores:**

**importar espacio**

**pnl spacy.load('es')**

**doc nlp(u'He volado a Los Ángeles. Ahora estoy volando a Frisco.')**

**para token en doc:**

**si token.ent\_type 1 0:**

**imprimir(token.text, token.ent\_type\_)**

**Si el atributo de tipo ent de un token no está establecido en 00, entonces el token es una entidad con nombre. Si es así, imprime el ent\_type\_attribute de un token, que contiene el tipo de entidad nombrada en Unicode. Como resultado, el script debería generar lo siguiente:**

**LA GPE**

**Frisco GPE**

**Tanto Los Ángeles como Frisco están marcados como GPE, el acrónimo de "entidad geopolítica" e incluye países, ciudades, estados y otros nombres de lugares.**

**maria**

**En este capítulo, configurará un entorno de trabajo para usar spaCy. Luego, aprendió scripts simples que ilustran cómo usar las funciones de spaCy para realizar las operaciones básicas de PNL para extraer información importante. Estas operaciones incluían tokenización, lematización e identificación de relaciones sintácticas entre tokens individuales en una oración. Los ejemplos proporcionados en este capítulo están simplificados y no reflejan escenarios del mundo real. Para escribir un script más sofisticado usando spaCy, necesitará implementar un algoritmo para derivar los tokens necesarios de un árbol de dependencia, usando las características lingüísticas asignadas a los tokens. Volveremos a extraer y utilizar características lingüísticas en el Capítulo 4 y cubriremos los árboles de dependencia en detalle en el Capítulo 6.**

**En el próximo capítulo, veremos los objetos clave de la API de spaCy, incluidos los contenedores y los componentes de la canalización de procesamiento. Además, aprenderá a utilizar las estructuras e interfaces de datos de nivel C de spaCy para crear módulos Python capaces de procesar grandes cantidades de texto.**

**La niebla**

**29**

### 2.2.6. La Biblia

La base teórica para desarrollar este punto es extensa y a veces muy controversial, pero se planteará el ser lo más objetivos posibles, estudiando libros de historiadores, teólogos, pastores de iglesias, y revisando documentos que nos den la base en cuanto a tener un texto que sea aceptado en la mayoría.

La palabra “Biblia” tiene raíz del latín medieval y deriva del griego byblos, que significa “libros”, pero si vamos hacia su origen etimológico del término llegamos a que byblos era una antigua ciudad Fenicia situada sobre las costas del actual Líbano. Los fenicios inventaron el alfabeto que todavía utilizamos y les enseñaron a escribir a los griegos. Desde Biblos los fenicios exportaban los papiros en los que fueron escritos los primeros libros (El papiro una planta semejante al junco era abierta en tiras que se humedecían y entretejían. Una vez secas, constituían un excelente papel en que escribir.) Aunque byblos originalmente significa “papiro” en griego, con el tiempo paso a significar “libros” Los libros \*tal como lo conocemos deben su denominación a la antigua ciudad. (Davis, 1998).

Ahora bien, el termino no es solo un libro sino la recopilación de muchos libros que fueron escritos en un tiempo aproximado de 4000 años por diferentes autores que trataron temas de leyes, poesía, filosofía e historia sabiduría. La reunión y organización de la biblia también se vuelve en un tema de estudio interesante, ya que dependiendo del grupo religioso que se esté basado su organización cambia, por ejemplo, la biblia de un judío, no es la misma de un católico y la biblia de un católico no es la de un protestante.

La Biblia se divide en Antiguo Testamento y Nuevo Testamento. Es interesante preguntarse a que se refiere la palabra testamento, que tiene varios significados. Uno es algo que mucho no quisiéramos pensar y es la última voluntad que es un documento legal para el uso de los bienes terrenales de un difunto. Otro significado es el dejar evidencia de algo y la utilizada para aludir a nuestro tema en cuestión es la manera antigua de decir “pacto” que aludía a un acuerdo o contrato. O sea, el antiguo testamento era el pacto entre Dios y su pueblo y con el nuevo testamento es un nuevo pacto a través de la vida, la muerte y la resurrección de Jesús.

**Antiguo Testamento**

Para los judíos que no tienen sino el antiguo testamento ya que no creen en el nuevo testamento escrita hebreo antiguo se puede decir que es el equivalente al antiguo testamento de la biblia de los cristianos. Se divide en tres: Tora, Profetas y escrituras

|  |  |  |
| --- | --- | --- |
| **Tora** | **Profetas** | **Escrituras** |
| Génesis  Éxodo  Levítico  Números  Deuteronomio | Jesúa  Jueces  Samuel (I)  Samuel (II)  Reyes (I)  Reyes (II)  Isaías  Jeremías  Ezequiel  Oseas  Joel  Amos  Abdías  Jonás  Miqueas  Nahum  Habacuc  Sofonías  Hageo  Zacarias  Malaquías | Salmos  Proverbios  Job  Cantar de Los cantares  Ruth  Lamentaciones  Eclesiastés  Ester  Daniel  Esdras  Nehemías  Crónicas (I)  Crónicas (II) |

Tabla 3 Libros de la Biblia Hebrea o Antiguo testamento

La biblia para la mayoría de los cristianos antiguo testamento. Los 39 libros del antiguo testamento se clasifican en cuatro grupos de acuerdo a la misma clase de escritura:

|  |  |
| --- | --- |
| **Grupo** | **Libros mayoría iglesias cristianas** |
| Ley | Génesis  Éxodo  Levítico  Números  Deuteronomio |
| Historicos | Josué  Jueces  Ruth  Samuel (I)  Samuel (II)  Reyes (I)  Reyes (II)  Crónicas (I)  Crónicas (II)  Esdras  Nehemías  Ester |
| Poesia | Job  Salmos  Proverbios  Eclesiastés  Cantares |
| Profetas | Isaías  Jeremías  Lamentaciones  Ezequiel  Daniel  Oseas  Joel  Amos  Abdías  Jonás  Miqueas  Nahum  Habacuc  Sofonías  Hageo  Zacarias  Malaquías |

Tabla 4 Libros Antiguo testamento mayoría biblias cristianas

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| ***Tanaj* (Biblia judía) (24 libros) ​ Los libros en negrita forman parte del *Ketuvim*** | **Antiguo Testamento Protestante (39 libros)** | **Antiguo Testamento Iglesia católica (46 libros)** | **Antiguo Testamento Iglesia ortodoxa (51 libros)** | **Idioma original** |
| *Torah* | *Pentateuco* o los *Cinco Libros de Moisés* | | |  |
| Bereishit | Génesis | Génesis | Génesis | Hebreo |
| Shemot | Éxodo | Éxodo | Éxodo | Hebreo |
| Vayikra | Levítico | Levítico | Levítico | Hebreo |
| Bamidbar | Números | Números | Números | Hebreo |
| Devarim | Deuteronomio | Deuteronomio | Deuteronomio | Hebreo |
| *Nevi'im (Profetas)* | *Libros históricos* | | |  |
| Yehoshua | Josué | Josué | Josué (Iesous) | Hebreo |
| Shofetim | Jueces | Jueces | Jueces | Hebreo |
| **Rut (Ruth)** | Rut | Rut | Rut | Hebreo |
| Shemuel | 1 Samuel | 1 Samuel (1 Reyes) ​ | 1 Samuel (1 Reinos) ​ | Hebreo |
| 2 Samuel | 2 Samuel (2 Reyes) ​ | 2 Samuel (2 Reinos) ​ | Hebreo |
| Melakhim | 1 Reyes | 1 Reyes (3 Reyes) ​ | 1 Reyes (3 Reinos) | Hebreo |
| 2 Reyes | 2 Reyes (4 Reyes)6​ | 2 Reyes (4 Reinos)7​ | Hebreo |
| **Divrei Hayamim (Crónicas)** ​ | 1 Crónicas | 1 Crónicas (1 Paralipómenos) | 1 Crónicas (1 Paralipómenos) | Hebreo |
| 2 Crónicas | 2 Crónicas (2 Paralipómenos) | 2 Crónicas (2 Paralipómenos) | Hebreo |
|  |  |  | 1 Esdras | Hebreo |
| **Ezra-Nehemiah** | Esdras | Esdras (1 Esdras) | Esdras (2 Esdras) ​ | Hebreo y Arameo |
| Nehemías | Nehemías (2 Esdras) | Nehemías (2 Esdras)​ | Hebreo |
|  |  | Tobit (Tobias) | Tobit (Tobias) | Arameo (¿y Hebreo?) |
| Judith | Judith | Hebreo |
| **Esther**5​ | Esther | Esther | Esther | Hebreo |
|  |  | 1 Macabeos | I Macabeos | Hebreo |
| 2 Macabeos​ | II Macabeos | Griego |
|  | III Macabeos | Griego |
| IV Macabeos | Griego |
| *Ketuvim (Escritos)* | *Libros sapienciales* | | |  |
| **Iyov (Job)**5​ | Job | Job | Job | Hebreo |
| **Tehillim (Salmos)**5​ | Salmos | Salmos | Salmos | Hebreo |
|  |  |  | Oración de Manasés | Griego |
| **Mishlei (Proverbios)**5​ | Proverbios | Proverbios | Proverbios | Hebreo |
| **Qoheleth (Eclesiastés)**5​ | Eclesiastés | Eclesiastés | Eclesiastés | Hebreo |
| **Shir Hashirim (Cantar de los Cantares)**​ | Cantar de Salomón | Cantar de los Cantares | Cantar de los Cantares (Aisma Aismaton) | Hebreo |
|  |  | Sabiduría | Sabiduría | Griego |
| Sirach (Eclesiástico) | Sirach | Hebreo |
| *Nevi'im (Últimos Profetas)* | *Profetas mayores* | | |  |
| Yeshayahu | Isaías | Isaías | Isaías | Hebreo |
| Yirmeyahu | Jeremías | Jeremías | Jeremías | Hebreo y Arameo |
| **Eikhah (Lamentaciones)** | Lamentaciones | Lamentaciones | Lamentaciones | Hebreo |
|  |  | Baruc | Baruc | Hebreo​ |
| Carta de Jeremías​ | Griego (opinión mayoritaria) |
| Yekhezqel | Ezequiel | Ezequiel | Ezequiel | Hebreo |
| **Daniel**5​ | Daniel | Daniel​ | Daniel | Hebreo y Arameo |
|  | *Profetas menores* | | |  |
| Los Doce o *Trei Asar* | Oseas | Oseas | Oseas | Hebreo |
| Joel | Joel | Joel | Hebreo |
| Amós | Amós | Amós | Hebreo |
| Abdías | Abdías | Abdías | Hebreo |
| Jonás | Jonás | Jonás | Hebreo |
| Miqueas | Miqueas | Miqueas | Hebreo |
| Nahum | Nahum | Nahum | Hebreo |
| Habacuc | Habacuc | Habacuc | Hebreo |
| Sofonías | Sofonías | Sofonías | Hebreo |
| Hageo | Hageo | Hageo | Hebreo |
| Zacarías | Zacarías | Zacarías | Hebreo |
| Malaquías | Malaquías | Malaquías | Hebreo |

Tabla 5 Antiguo testamento (Antiguo Testamento, n.d.)

Otro punto que de los autores hay controversias pero que en los mismos escritos se pueden extractar quien es el que los escribe, aunque si hay libros que siguen con diferentes teorías de quien es el autor, Un ejemplo es Deuteronomio que se le considera a Moisés el autor, pero en el último capítulo data su muerte, por lo que probablemente este fue escrito por Josué. (Jesúa en el Tora)

**El nuevo testamento**

El nuevo testamento es la segunda parte de la biblia cristiana. El nuevo testamento narra la vida, ministerio, crucifixión y resurrección de Jesucristo, así como los eventos del cristianismo del siglo primero. Fue compuesto entre los años 50 y 100 d.C.

Las versiones antiguas de los textos del nuevo testamento están escritas en el griego denominado Koiné, lengua franca en el mediterráneo oriental en época romana. Aunque algunos escritos pueden haberse escrito primeramente en hebreo o arameo, la lengua semita hablada por Jesús y su entorno (Alvear, 1995). Aún hoy existen textos manuscritos fechados como desde el siglo v (cercanos a los más antiguos manuscritos griegos completos) en arameo como la Peshita siríaca, la Harclense y la Curetoniana, pero la mayoría de los estudiosos los consideran traducciones del griego.

El Nuevo Testamento comprende los cuatro [evangelios canónicos](https://es.wikipedia.org/wiki/Evangelios_can%C3%B3nicos), los [Hechos de los Apóstoles](https://es.wikipedia.org/wiki/Hechos_de_los_Ap%C3%B3stoles), las [epístolas](https://es.wikipedia.org/wiki/Ep%C3%ADstolas_paulinas) de [Pablo de Tarso](https://es.wikipedia.org/wiki/Pablo_de_Tarso), siete [epístolas católicas](https://es.wikipedia.org/wiki/Ep%C3%ADstolas_cat%C3%B3licas) de diversa atribución y el [Apocalipsis](https://es.wikipedia.org/wiki/Apocalipsis), como se puede observar en el esquema que se encuentra a continuación.

Comprende, en total, 27 libros en el [canon](https://es.wikipedia.org/wiki/Canon_b%C3%ADblico_de_la_Iglesia_cat%C3%B3lica) aceptado por la mayoría de las Iglesias. La Iglesia Siria solo acepta 22 libros en su canon. Libros como 1 y 2 de Clemente, el libro de la Alianza, el Octateuco y otros, han sido motivo de disputas.

| **Orden** | **Libro** | **Abreviatura** | **N.º Capítulos** |
| --- | --- | --- | --- |
| 1 | Evangelio de Mateo | Mt | 28 |
| 2 | Evangelio de Marcos | Mc | 16 |
| 3 | Evangelio de Lucas | Lc | 24 |
| 4 | Evangelio de Juan | Jn | 21 |
| 5 | Hechos de los Apóstoles | Hch | 28 |
| 6 | Epístola a los romanos | Rom | 16 |
| 7 | Primera epístola a los corintios | 1 Cor | 16 |
| 8 | Segunda epístola a los corintios | 2 Cor | 13 |
| 9 | Epístola a los gálatas | Gal | 6 |
| 10 | Epístola a los efesios | Ef | 6 |
| 11 | Epístola a los filipenses | Flp | 4 |
| 12 | Epístola a los colosenses | Col. | 4 |
| 13 | Primera epístola a los tesalonicenses | 1 Ts | 5 |
| 14 | Segunda epístola a los tesalonicenses | 2 Ts | 3 |
| 15 | Primera epístola a Timoteo | 1 Tim | 6 |
| 16 | Segunda epístola a Timoteo | 2 Tim | 4 |
| 17 | Epístola a Tito | Tit | 3 |
| 18 | Epístola a Filemón | Flm | 1 |
| 19 | Epístola a los hebreos | Heb | 13 |
| 20 | Epístola de Santiago | Sto | 5 |
| 21 | Primera epístola de Pedro | 1 P | 5 |
| 22 | Segunda epístola de Pedro | 2 P | 3 |
| 23 | Primera epístola de Juan | 1 Jn | 5 |
| 24 | Segunda epístola de Juan | 2 Jn | 1 |
| 25 | Tercera epístola de Juan | 3 Jn | 1 |
| 26 | Epístola de Judas | Jud. | 1 |
| 27 | Apocalipsis | Ap | 22 |

Tabla 6 Libros del nuevo testamento más aceptaods (Nuevo Testamento, n.d.)

La composición del canon del nuevo testamento data desde los años 170 DC.

Los manuscritos del nuevo testamento según Robert W. Funk, fundador del [Jesus Seminar](https://es.wikipedia.org/wiki/Jesus_Seminar) (‘seminario de Jesús’), existen muchas variantes en los distintos manuscritos griegos del Nuevo Testamento que han llegado hasta la actualidad; algunas son variantes menores sin trascendencia, pero también hay cambios significativos.

Se ha estimado que hay más de 70.000 variantes significativas en los manuscritos griegos del Nuevo Testamento. Tal montaña de variaciones ha sido reducida a un número manejable por las ediciones críticas modernas que ordenan, evalúan y eligen entre la mirada de posibilidades. Las ediciones críticas del Nuevo Testamento griego utilizadas por eruditos son, de hecho, creaciones de los críticos textuales y editores. No son idénticas a ninguno de los manuscritos antiguos sobrevivientes. Son una composición de muchas versiones distintas.

### 2.2.7. Dataset bíblicas

Se han buscado dataset públicos bíblicos para entrenar y generar análisis texto relacionado con la biblia. Se encuentran versiones digitales de la biblia en diferentes formatos y sitios web.

Un primer sitio consultado es un sitio web dedicado a la cronología bíblica realizada por Rick Aschman la cual tiene un estudio que presenta datos y una visualización que nos da una orientación hacia el uso de datos para presentar eventos, personajes y fechas con una visualización en una línea de tiempo. (Aschmann, 2022)

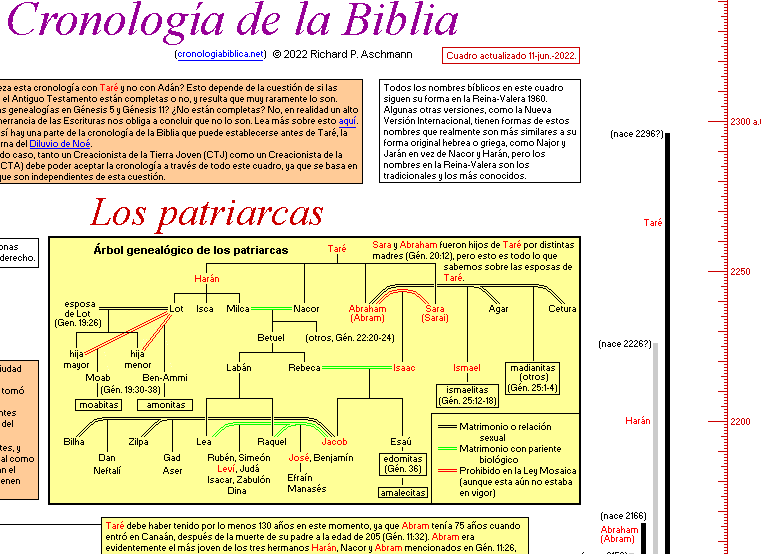


Figura 12 Parte de la visualización de cronología de la Biblia (Aschmann, 2022)

Otras datasets de biblias las encontramos en los corpus por ejemplo en los repositorios de GitHub que al ser abiertas se pueden investigar para trabajarlo en el prototipo son proyectos enfocados a hacer traducciones de la biblia en diferentes idiomas (Christodoulopoulos, n.d.)

## 2.3 Conclusiones (nexo de unión de lo investigado con el trabajo a realizar)

* Se está trabajando un grupo de pastores en Colombia que serán parte del proceso de analizar las dificultades y serán parte de la investigación, por medio de entrevistas y soporte en el prototipo de parte del uso.

# 3. Objetivos concretos y metodología de trabajo

## 3.1. Objetivo general

El objetivo general de este piloto experimental es desarrollar un sistema de análisis de datos masivos y visualización que facilite la comprensión de la narrativa bíblica en los contextos históricos de tiempo y cada momento de la historia.

## 3.2. Objetivos específicos

* Desarrollar un prototipo de modelo NLP que permita extraer información cronológica de los textos bíblicos.
* Crear una base estructurada de las fechas de los personajes y eventos bíblicos.
* Utilizar herramientas de visualización para presentar los eventos y personajes en forma dinámica.

# 4. Desarrollo específico de la contribución

## 4.1. Descripción detallada del Piloto experimental

Este piloto experimental es controversial porque cuando se habla de “big data” se refiere a conjuntos de datos grandes y complejos cuyos procesamientos de datos no encuentran herramientas óptimas para manejarlos eficientemente. Al tomar la Biblia como un texto de datos a analizar, por ser la Biblia una colección de textos religiosos, puede considerarse como un conjunto de datos, pero su clasificación como big data dependerá de la escala y el contexto de comparación.

Algunos puntos a considerar están, en primer lugar, el volumen de datos donde la Biblia contiene un texto considerable como se expuso anteriormente, pero comparándola con algunos datos que manejan las redes sociales, los registros de las transacciones financieras, datos de sensores, puede ser que la Biblia no sea tan grande.

Otro punto es la complejidad del texto donde su lingüística y su valor cultural e histórico la hacen significativamente para el ambiente del procesamiento de lenguaje natural lo que conlleva a plantear desafíos y requerir técnicas avanzada de análisis de texto y aun más cuando se realiza un análisis detallado se necesita de enfoques y herramientas de procesamiento de grandes volúmenes de texto.

### 4.1.1 Qué tecnologías se utilizaron (incluyendo justificación de por qué se emplearon y descripciones detalladas de las mismas).

Las tecnologías utilizadas en el proyecto son NLP, Python spaCy, Google Colab, GitHUb

Las Datasets analizadas están en internet, hay proyectos que iniciaro y están sin terminar o sin actualizar por años. También las estructuras son diferentes y con tecnologías diferentes lo que lleva a hacer transformación de datos para poder utilizar los datasets, en los cuales hay que pedir autorizaciones y no están tan accesibles para trabajarlas en el prototipo.

### 4.1.2 Cómo se organizó el piloto

**Colaboración compartir código**

El piloto se desarrolla en un tema muy específico que va a necesitar una colaboración eficiente de un grupo de trabajo que participe colocando y alimentando datos en el tiempo. Por esto se organiza el proyecto en un repositorio de GitHub, que facilita el trabajo en conjunto, el seguimiento de cambios y la gestión de versiones del código.

**Accesibilidad y uso de la nube.**

Con Google Colab se ofrece el entorno de desarrollo basado en las herramientas de Júpiter Notebooks. También facilita el acceso y la ejecución de código solo teniendo una conexión de internet y además que ya el entorno de desarrollo está en la nube con la ventaja de tener las herramientas ya instaladas en la nube.

**Reproducibilidad del proyecto**

GitHub gestiona las dependencias del proyecto. En los archivos requiements.txt especifica las bibliotecas y versiones necesarias para ejecutar el código. Google Colab permite que se instalen las dependencias directamente desde el cuaderno.

**Uso de spaCy y NLP**

Organizando el código en un repositorio facilita la integración de spaCy en el flujo de trabajo y permite la implementación clara de las tareas específicas de NLP relacionada con la extracción de información de los textos bíblicos con fines a crear visualizaciones de cronología de persones y/o eventos.

**Seguridad y Control de acceso:**

GitHub ofrece opciones de seguridad y control de acceso. Se puede configurar el repositorio privado, limitando el acceso a los colaboradores autorizados. Esto es importante para proteger datos sensibles y el código del proyecto.

**Documentación clara y detallada:**

En el archivo README se proporciona información detallada de como configurar el entorno y entender el entorno del proyecto.

**Integración con servicio de Despliegue y continua integración**

En este proyecto es crear la plataforma para que servicios de Integración continua y despliegue del software para mejorar la eficiencia, la calidad y la velocidad del ciclo de vida del desarrollo del software.

### 4.1.3 Qué personas participaron (con datos demográficos) o qué técnicas de evaluación automática se emplearon.

El grupo de personas al cual esta enfocado son expertos de teología como pastores, teólogos académicos con experiencia en estudios bíblicos y a usuarios finales que en si son miembros de congregaciones y comunidades cristianas.

La Federación Bautista independiente de Colombia y sus miembros son el grupo piloto para hacer las evaluaciones del prototipo. Los pastores que tienen años de experiencia y con estudios teológicos y están en Bogotá primordialmente y áreas cercanas

### 4.1.4. Cómo transcurrió el experimento.

* Lo primero fue crear el sitio de GitHub
* Crear el sitio de Google Colab
* Se clono el repositorio que esta en GitHub en Google Colab y así se descarga el contenido.

Python:

Traer un repositorio a Google Colab desde GitHub (Anexo2)

Las bases de datos necesitan transformaciones. Para trabajar en SQL con bases de datos que vienen de GitHub, hay algunos cambios a realizar en líneas de código, Los

* Se hacen ejemplos con personajes y eventos de

### 4.1.5 Qué instrumentos de seguimiento y evaluación se utilizaron.

Qué tipo de análisis estadísticos se ha empleado (si procede).

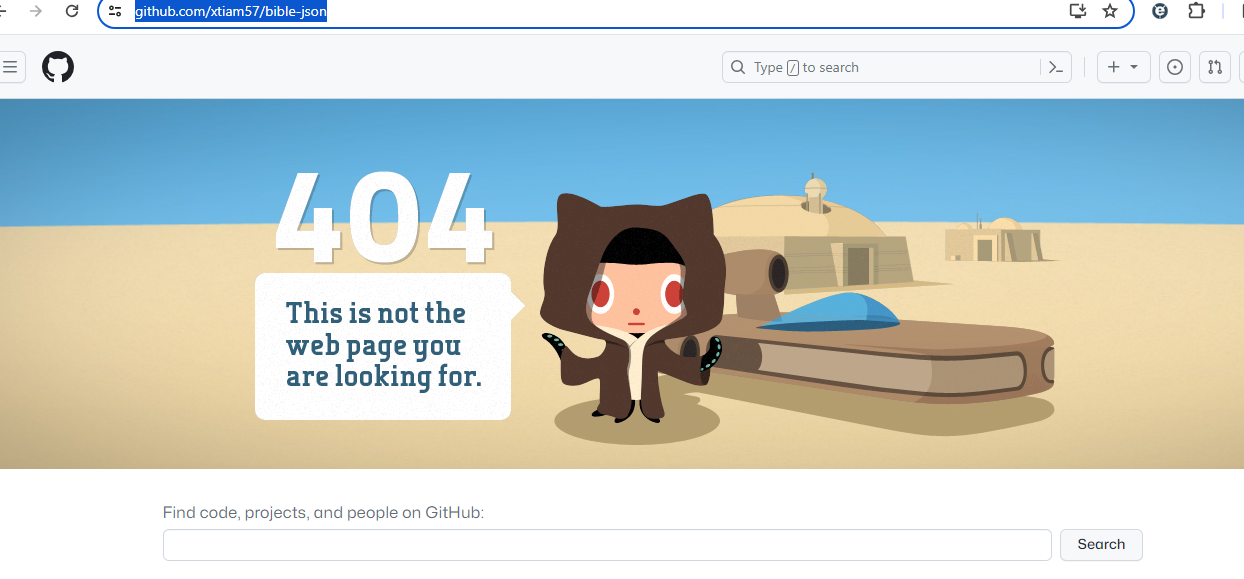
**Para la evaluación de los resultados se diseñan encuentas.**

## 4.2 Descripción de los resultados

En la evaluación de resultados se tienes varios puntos”

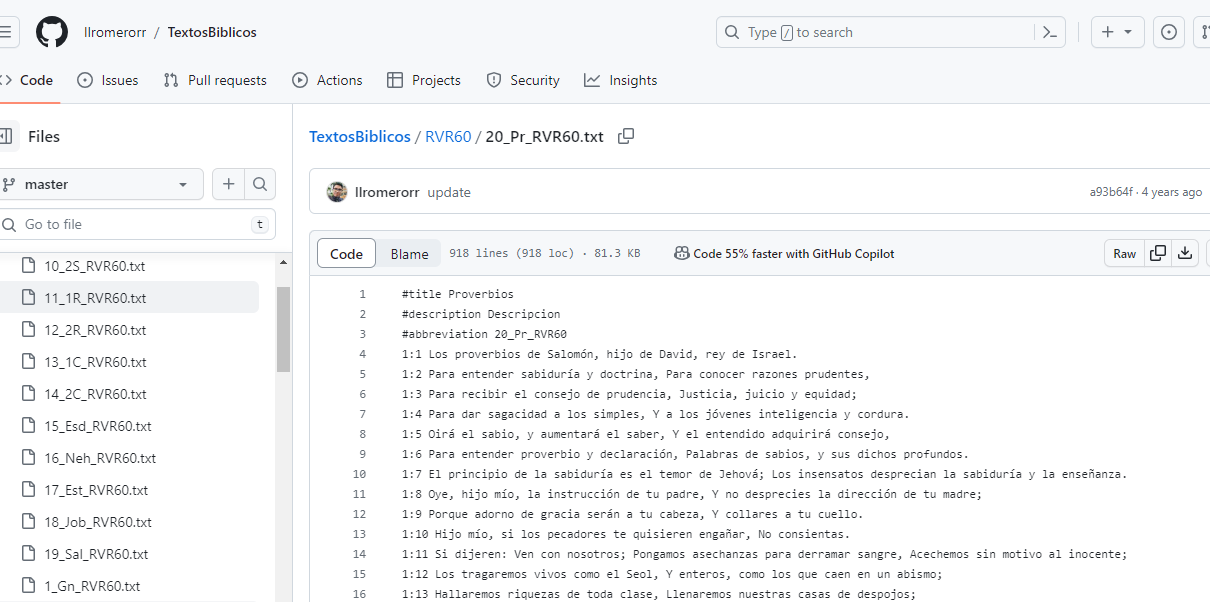
**Evaluación de los datasets**

En la búsqueda de datasets se encuentran varios sitios que empezaron hace varios años y no están disponibles, otros ya no están en las paginas que se revisaron:



<https://github.com/xtiam57/bible-json>

Otros dataset tienen estructuras diferentes que hay que transformar o unir datos de texto en una solo dataset



<https://github.com/llromerorr/TextosBiblicos/blob/master/RVR60/20_Pr_RVR60.txt>

Para leer las dataset de texto desde Google Colab se utiliza la biblioteca ***request***

El código para leer este estilo de archivos esta en anexo 2.

Evaluación de los repositorios

Evaluación de las bases de datos de entrenamiento y prueba

Evaluación de los resultados de eventos y personajes.

## 4.3. Discusión

Tras la presentación objetiva de los resultados, querrás aportar una discusión de los mismos. En este capítulo puedes discutir la relevancia de los resultados, presentar posibles explicaciones para los datos anómalos y resaltar aquellos datos que sean particularmente relevantes para el análisis del experimento.

El piloto experimental propuesto enfocado a contribuir …….

En este capítulo debes desarrollar la descripción de tu contribución. Es muy dependiente del tipo de trabajo concreto, y puedes contar con la ayuda de tu director para estudiar cómo comunicar los detalles de tu contribución.

# 5. Conclusiones y trabajo futuro

## 5.1. Conclusiones

Este último bloque es habitual en todos los tipos de trabajos y presenta el resumen final de tu trabajo y debe servir para informar del alcance y relevancia de tu aportación.

Suele estructurarse empezando con un resumen del problema tratado, de cómo se ha abordado y de por qué la solución sería válida.

Es recomendable que incluya también un resumen de las contribuciones del trabajo, en el que relaciones las contribuciones y los resultados obtenidos con los objetivos que habías planteado para el trabajo, discutiendo hasta qué punto has conseguido resolver los objetivos planteados.

## 5.2. Líneas de trabajo futuro

Finalmente, se suele dedicar una última sección a hablar de líneas de trabajo futuro que podrían aportar valor añadido al TFM realizado. La sección debería señalar las perspectivas de futuro que abre el trabajo desarrollado para el campo de estudio definido. En el fondo, debes justificar de qué modo puede emplearse la aportación que has desarrollado y en qué campos.

# 

# Bibliografía

*A webcomic of romance,*. (n.d.). Retrieved from vuelve a golpear el clavo con el martillo

Alvear, J. (1995). Cristianismo Primitivo y Religiones Mistéricas. In J. Alvear, *Cristianismo Primitivo y Religiones Mistéricas* (p. 19). Madrid: Ed. Cátedra. Historia Serie Mayor.

ASALE, R. &. (n.d.). *Real Academia Española*. Retrieved from https://dle.rae.es/inteligencia

Aschmann, R. (2022, 05 21). *Chronology of the Bible*. Retrieved from https://aschmann.net/BibleChronology/castellano.html

*Azure AI Language*. (n.d.). (Microsoft, Producer) Retrieved 1 3, 2024, from https://azure.microsoft.com/en-us/services/cognitive-services/text-analytics/

Bramlett, T. (2016, 12 7). *Strings, Bytes, and Unicode in Python 2 and 3*. Retrieved from https://timothybramlett.com/Strings\_Bytes\_and\_Unicode\_in\_Python\_2\_and\_3.html

Center, t. G. (n.d.). *Latest Westminster Leningrad Codex (WLC) Is Available on the Web*. Retrieved from https://www.grovescenter.org/latest-westminster-leningrad-codex-wlc-is-available-on-the-web/

Cheesem, P. (n.d.). *EEtimes*. Retrieved from https://www.eetimes.com/digital-data-storage-is-undergoing-mind-boggling-growth/

Coghlan, N. (2014, 01 06). *Python notes*. Retrieved 2024, from https://python-notes.curiousefficiency.org/en/latest/python3/binary\_protocols.html

*Concepto y definicion Net*. (2023, 7 28). Retrieved from https://conceptodefinicion.net/descubre-cuantas-palabras-hay-en-la-biblia/

CORPUS, O. O. (n.d.). *anc American National Corpus*. Retrieved 12 01, 2023, from www.anc.org

David Cournapeau, M. B. (n.d.). *scikit-learn*. Retrieved from Machine Learning in Python: https://scikit-learn.org/stable/

Davis, K. C. (1998). *Que se yo de la Biblia.* (T. Arijon, Trans.) New York: Sudamericana.

*EL orden Mundial EOM*. (2019, 12 20). Retrieved from El cristianismo en el mundo: https://elordenmundial.com/mapas-y-graficos/cristianismo-en-el-mundo/

*Embajada de Israel en España*. (s.f.). Retrieved from https://embassies.gov.il/madrid/AboutIsrael/AmongtheNations/Pages/ENTRE-NACIONES-Judeidad.aspx

*GENSIM topic modeling for humans*. (Consultada en Nov 22 2023). Retrieved from https://radimrehurek.com/gensim/

Hutchins, J. (1999). Retrospect and prospect in computer-based translation., (pp. 31-34). Retrieved from www.nt-archive.info/90/MTS-1999-Hutchins.pdf

*List of text corpora*. (n.d.). Retrieved 12 01, 2023, from https://en.wikipedia.org/wiki/List\_of\_text\_corpora

*List of text corpora*. (2023, 11 22). Retrieved from List of text corpora: https://en.wikipedia.org/wiki/List\_of\_text\_corpora

Liviu P. Dinu, V. N.-O. (2012, 04 05). *ACL Anthology*.

L-uhn, H. (1958, 10 1). *ACM DL DIGITAL LIBRARY*. Retrieved 12 1, 2023, from https://dl.acm.org/doi/10.1147/rd.24.0314

mySociety. (n.d.). *They Work For You*. Retrieved from https://www.theyworkforyou.com/

*NLTK*. (Consultada en 2023, 11 22). Retrieved from https://www.nltk.org/

*NVIDIA DEVELOPER*. (2015, JUN 14). Retrieved from https://developer.nvidia.com/blog/introduction-neural-machine-translation-gpus-part-2/

*Python - Built-in types*. (n.d.). Retrieved 2004, from https://docs.python.org/3/library/stdtypes.html

*Python library for pulling data out of HTML and XML files*. (n.d.). Retrieved from Beautiful Soup Documentation: https://www.crummy.com/software/BeautifulSoup/bs4/doc/

*Python string common operations*. (n.d.). Retrieved 2004, from https://docs.python.org/3/library/string.html

Python. (n.d.). *UrlLib*. Retrieved from https://docs.python.org/3/library/urllib.html

Ravichandiran, S. (January 2021). *Getting Started with Goggle BERT.* Birmingham, UK: Packt Publishing Ltda.

*reddit* . (n.d.). Retrieved from List of text corpora

RoBERTa, s. (2020). *spaCy*. Retrieved from Natural Language: https://spacy.io/

Ronacher, A. (2014, 01 5). *Armin Ronacher's Thoughts and writings*. Retrieved from https://lucumr.pocoo.org/2014/1/5/unicode-in-2-and-3/

Sarah Guthals, P. P. (2019). *GitHub dummies A wiley brand.* Hooboken, NJ, USA: John Wiley & Son, inc.

SBLGNT. (n.d.). *Society of Biblical literture* . Retrieved from https://www.sblgnt.com/

*Scrapy*. (n.d.). Retrieved 01 02, 2024, from https://scrapy.org/

*Social Media Sentiment Visualization*. (n.d.). Retrieved from Social Media Sentiment Visualization: https://www.csc2.ncsu.edu/faculty/healey/social-media-viz/production/

Sowmya Vajjala, B. M. (2020). *Practical Natural language Processing.* Sebastopol, CA, USA: O'Relly Media, inc.

Srinivasa-Desikan, B. (2018). In B. Srinivasa-Desikan, *Natural Language Processing and Computacional Linguistics* (pp. 9-23).

TensorFlow. (n.d.). *Create production-grade machine learning models with TensorFlow*. Retrieved from https://www.tensorflow.org/

*Textblob: Simplified Text Processing*. (Consultado en 2023, Noviembre). Retrieved from https://textblob.readthedocs.io/en/dev/

*The British National Corpus (BNC)*. (n.d.). Retrieved Enero 4, 2004, from http://www.natcorp.ox.ac.uk/corpus/index.xml

Thiyagarajan, S. (2018, 01). High user Experience by Providing Relevant News Articles using Topic Modelling. *International Journal of Engineering Trends and Technology (IJETT) , 55*(1). Retrieved from https://www.researchgate.net/publication/323609994\_High\_user\_Experience\_by\_Providing\_Relevant\_News\_Articles\_using\_Topic\_Modelling

Today, P. (2023). *Shakespeare and his co-authors, as told by Penn engineers*. Retrieved from https://penntoday.upenn.edu/spotlights/shakespeare-and-his-co-authors-told-penn-engineers

*Tranformers Huggin Face*. (Consultado en nov 22 2023). Retrieved from https://huggingface.co/docs/transformers/index

*Tweepy*. (n.d.). Retrieved from https://www.tweepy.org/

Vasiliev, Y. (2020). *Natural Language Proccesing with Python and Spacy A practical introduction.* San Francisco, CA, USA: No starch Press, Inc.

Whatsapp Preguntas. (n.d.). *Help Center Whatsapp*. Retrieved from https://faq.whatsapp.com/1180414079177245/?locale=en\_US&cms\_platform=android

*Wikipedia*. (n.d.). Retrieved 01 02, 2024, from Wikipedia: https://es.wikipedia.org/wiki/Nuevo\_Testamento

*Wikipedia*. (n.d.). Retrieved 01 02, 2024, from Wikipedia: https://es.wikipedia.org/wiki/Antiguo\_Testamento

# Anexos

## Anexo I. Entrevista 1

Se realizo la encuesta utilizando Google Forms

<https://docs.google.com/forms/d/1wJe3Jz9xDQ5TeTRcB6Xhg-S9VGoPcdM5bJ2dxMD6gCk/viewanalytics>

Estimado Pastor

Es un placer para nosotros invitarlo participar en esta encuesta que forma parte de un proyecto de investigación centrado en enriquecer el estudio de la Biblia y las enseñanzas pastorales mediante herramientas de visualización. Su valiosa perspectiva como pastor cristiano será fundamental para comprender mejor cómo las herramientas visuales pueden ser de ayuda en el análisis y la presentación de la cronología de personajes y eventos bíblicos.

Agradecemos sinceramente su participación. Todas las respuestas serán confidenciales y se utilizarán únicamente con fines de investigación académica.

\*\*Encuesta: \*\*

1. \*\*¿Cuántos años de experiencia tienes como pastor cristiano? \*\*

- Menos de 1 año

- 1-5 años

- 6-10 años

- Más de 10 años

2. \*\*¿En qué denominación o tradición cristiana sirves como pastor? \*\*

- [Espacio para respuesta abierta]

3. \*\*¿Con qué frecuencia consultas la Biblia en tus enseñanzas y sermones? \*\*

- Diariamente

- Semanalmente

- Mensualmente

- Ocasionalmente

4. \*\*¿Cuáles son tus libros bíblicos favoritos para enseñar y por qué? \*\*

- [Espacio para respuesta abierta]

5. \*\*¿Utilizas herramientas de visualización para ayudarte a comprender mejor la Biblia y preparar tus enseñanzas? \*\*

- Sí

- No

- No estoy seguro

6. \*\*¿Qué tipo de visualizaciones te gustaría tener para entender la cronología de personajes y eventos bíblicos? \*\*

- Gráficos de línea temporal

- Diagramas de flujo

- Otros (especifica)

7. \*\*¿Cuál es tu personaje bíblico favorito y por qué? \*\*

- [Espacio para respuesta abierta]

8. \*\*¿Hay algún evento bíblico en particular que encuentres especialmente significativo para tu ministerio? \*\*

- [Espacio para respuesta abierta]

9. \*\*¿Te gustaría profundizar en el estudio de algún personaje o evento bíblico específico utilizando herramientas de visualización? \*\*

- Sí

- No

- No estoy seguro

10. \*\*¿Cómo crees que la visualización de la cronología de personajes y eventos bíblicos podría enriquecer tus enseñanzas y tu ministerio pastoral? \*\*

- [Espacio para respuesta abierta]

11. \*\*¿Tienes alguna sugerencia o comentario adicional sobre cómo mejorar el estudio de la Biblia mediante herramientas de visualización? \*\*

- [Espacio para respuesta abierta]

Agradecemos sinceramente su participación y contribución a esta investigación.

## Anexo 2. Código

*Codigo para obtener la dataset base sql desde Google Colab*

*!git* clone <https://github.com/iglesianazaret/biblia-reina-valera-1909-base-datos-sql.git>

!git clone https://github.com/Davidgoca/biblia-reina-valera-1909-base-datos-sql

import sqlite3

# Conectar a la base de datos

conn = sqlite3.connect('/content/biblia-reina-valera-1909-base-datos-sql/reina\_valera\_1909.db')

# Crear un cursor

cursor = conn.cursor()

# Ejecutar una consulta para los primeros 20 versículos

cursor.execute('SELECT \* FROM verses LIMIT 20')

result = cursor.fetchall()

# Imprimir los resultados

for row in result:

print(row)

# Ejecutar una consulta para los primeros 20 libros

cursor.execute('SELECT \* FROM books LIMIT 20')

result = cursor.fetchall()

# Imprimir los resultados

for row in result:

print(row)

# Cerrar la conexión

conn.close()

## Anexo 3. Código

## *Federación Bautista independiente de Colombia.*

|  |  |
| --- | --- |
| ***IGLESIA*** | ***PASTOR*** |
| ***I.B. Betania.*** | ***Ptr. Daniel Cuellar*** |
| ***I.B. Betsaida*** | ***Ptr. Eduardo Cáceres*** |
| ***I.B El calvario Molinos*** | ***Ptr. Jairo Sanabria*** |
| ***I.B. Valle de Tenjo*** | ***Ptr. Omar Méndez*** |
| ***I.B. Emanuel Perdomo.*** | ***Ptr. Marco Orjuela*** |
| ***I.B. Esperanza viva. Suba.*** | ***Ptr Néstor Pinilla.*** |
| ***I.B. Getsemaní. Castilla.*** | ***Ptr Alex Abril*** |
| ***I.B. Gracia en abundacia*** | ***Ptr. Héctor López*** |
| ***I.B. Sumapaz*** | ***Ptr. Alejandro Morales*** |
| ***I.B Esperanza de Soacha.*** | ***Ptr. Julián Martínez*** |
| ***I.B. Impacto Bíblico.*** | ***Ptr. Jonathan Boyd*** |
| ***I.B. La fe.*** | ***Ptr. Anxander Castañeda*** |
| ***I.B. La gracia Chía.*** | ***Ptr. Galaxiux Castañeda*** |
| ***I.B. Emmaus.*** | ***Ptr Armando Bohórquez*** |
| ***I.B. Esperanza viva Medellín.*** | ***Ptr. Peter Hudson*** |
| ***I.B. La gracia. Villavicencio.*** | ***Ptr. Lisandro Vera*** |
| ***I.B. La misión.*** | ***Ptr. Ramón Perea*** |
| ***I.B. Valle de Josafat*** | ***Ptr. Mauricio Beltrán*** |
| ***I.B. Luz y verdad*** | ***Ptr. Timpthy Wheeler*** |
| ***I.B Torre Fuerte*** | ***Ptr. Andrés Pedraza*** |
| ***I.B. Maranatha*** | ***Ptr. Carlos Benites*** |
| ***I.B. Nueva Vida*** | ***Ptr. Wilson Moreno*** |
| ***I.B. Pontevedra*** | ***Ptr Joaquín Catalán*** |
| ***I.B. Tunal*** | ***Ptr. Daniel Patiño*** |
| ***I.B. Vida abundante. Envigado*** | ***Ptr Javier Ruiz*** |
| ***I.B. Ebenezer*** | ***Ptr. Leonardo Rozo*** |
| ***I.B. El Rosal*** | ***Ptr. Jesús David Pinzón*** |
| ***I.B. El Camino.*** | ***Ptr. Juan David Sánchez*** |
| ***I.B. Su gracia es mayor. Bosa*** | ***Ptr. Oscar Páez Triviño.*** |
| ***I.B. Piedra Angular.*** | ***Ptr. Ricardo Moreno.*** |