

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

Para ello se utilizan datasets públicos que contienen variables como libro, capítulo, versículo, “pasaje bíblico”, y mediante técnicas de **NLP** (Procesamiento de Lenguaje Natural) se analizan las variables y finalmente se crea una plataforma 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 utilizando plataformas colaborativas.

Lo que se espera de la investigación es un prototipo que utilizando el texto de la biblia permita usar 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, Python, spaCy, Neo4j

Abstract

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

To do this, public datasets are used that contain variables such as book, verse, "biblical passage", and using NLP (Natural Language Processing) techniques, the variables are analyzed and finally a platform is created that is accessible to any user who requires it.

For data collection, cleaning, purification, analysis, classification and visualization, Python will be used with its respective libraries using collaborative platforms.

What is expected from the research is 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, Python, spaCy. Neo4j

Índice de Contenido

| | |
|---|----|
| 1. Introducción..... | 8 |
| 1.1 Justificación | 8 |
| 1.2 Planteamiento del trabajo | 9 |
| 1.3 Estructura de la memoria..... | 9 |
| 2. Contexto y estado del arte..... | 10 |
| 2.1 Introducción al tema..... | 10 |
| 2.2. Desarrollo (base teórica): antecedentes, estudios actuales, autores de referencia | 10 |
| 2.2.1 Análisis de Texto | 11 |
| 2.2.2 NLP (Natural Language Processing) | 22 |
| 2.2.2.1 Word Embedding | 23 |
| 2.2.2.2 Pruebas | 29 |
| 2.2.2.3 Haciendo Predicciones | 30 |
| 2.2.3 Google BERT (Bidirectional Encoder Representations from Transformers) to | 31 |
| 2.2.5. Python..... | 31 |
| 2.2.5.1 Análisis de Texto..... | 31 |
| 2.2.5.2 Bibliotecas Python | 33 |
| 2.2.5.3 spaCy | 33 |
| 2.2.5.3.1 Tokenización | 34 |
| 2.2.5.3.2 Lematización | 35 |
| 2.2.5.3.3 Etiqueta | 38 |
| 2.2.4. Herramientas Colaborativas..... | 48 |
| 2.2.4.1 GitHub | 48 |
| 2.2.4.2 Google <i>Colab</i> | 49 |
| 2.2.6. Dataset en base de Datos de Grafos (Neo4j)..... | 50 |
| 2.2.7. La Biblia | 55 |
| 2.3 Conclusiones | 67 |
| 3. Objetivos concretos y metodología de trabajo | 67 |

| | |
|---|-----|
| 3.1. Objetivo general..... | 67 |
| 3.2. Objetivos específicos | 67 |
| 4. Desarrollo específico de la contribución | 68 |
| 4.1.1 Organización | 68 |
| 4.1.1.1. Repositorio..... | 68 |
| 4.1.1.2. Accesibilidad y uso de la nube. | 68 |
| 4.1.1.3 Evaluación Datasets | 69 |
| 4.1.1.4. Dataset Ideal..... | 70 |
| 4.1. Descripción piloto experimental | 76 |
| 4.1.3 Participación y técnicas de evaluación. | 77 |
| 4.1.4. Cómo transcurrió el experimento..... | 77 |
| 4.2 Descripción de los resultados | 78 |
| 4.3. Discusión | 80 |
| 5. Conclusiones y trabajo futuro | 81 |
| 5.1. Conclusiones | 81 |
| 5.2. Líneas de trabajo futuro | 82 |
| Anexos..... | 89 |
| Anexo I Encuestas..... | 89 |
| Anexo II Código de Python | 99 |
| Anexo III Federación Bautista independiente de Colombia. | 100 |
| Anexo IV Datasets | 101 |
| Anexo V Dataset Ideal | 106 |
| Anexo VI Modelo de base de datos de grafo | 107 |
| Anexo VII Código Cypher | 109 |

Índice de figuras

| | |
|--|-----------|
| Figura 1 Taxonomía del machine Learning dentro de la Inteligencia Artificial. Basado en Panesar, 2019 | 11 |
| Figura 2. Tasa de crecimiento de los datos entre 2006-2020 (Cheesem, n.d.)..... | 12 |
| 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 |
| Figura 4 Técnicas como el modelado de temas utilizan métodos de modelado probabilístico para identificar temas clave del texto. (Thiyagarajan, 2018)..... | 15 |
| Figura 5 Vuelve a golpear el clavo con el martillo (A webcomic of romance,, n.d.) | 19 |
| Figura 6 Un fragmento de una proyección de 2D | 26 |
| Figura 7 Generando un modelo estadístico con el algoritmo de aprendizaje automático | 29 |
| Figura 8 Ejemplo Predicción una estructura de árbol de dependencia usando un modelo estadístico | 30 |
| Figura 9 Vistazo General de los pasos de NLP | 34 |
| Figura 10 Ejemplo de código Python haciendo Tokenización en el análisis de texto... | 34 |
| Figura 11 Ejemplo de Lematización..... | 35 |
| Figura 12 Resultado del ejemplo Lematización | 36 |
| Figura 13 Usando Lematización en un requerimiento de un cliente con parte de la palabra | 36 |
| Figura 14 Ejemplo Código Python usando biblioteca Spacy para lematización | 37 |
| Figura 15 Código que genera los verbos relevantes de la oración. | 40 |
| Figura 16 Código para encontrar los verbos y los nombres propios..... | 41 |
| Figura 17 Una representación gráfica de un arco de dependencia sintáctica | 42 |
| Figura 18 Ejemplo representación gráfica de la dependencia sintáctica..... | 42 |

| | |
|---|-----------|
| Figura 19 Relaciones de Encabezado/hijo en una oración completa | 43 |
| Figura 20 Código para encontrar dependencia sintáctica | 46 |
| Figura 21 Código mostrando la raíz de la oración. | 46 |
| Figura 22 Código para identificar más dependencias y más entendimiento sintáctico. | 47 |
| Figura 23 Código reconocimiento de localidades geográficas..... | 48 |
| Figura 24 Esquema de Neo4j (Fuente: Tema 8 de clase Base de Datos para Big Data Universidad UNIR)..... | 52 |
| Figura 25 Representación gráfica de nodo y de vértice | 53 |
| Figura 26 Representación de relaciones mediante grafos..... | 53 |
| Figura 27 Componentes de Base de Grafos Neo4j..... | 54 |
| Figura 28 Estudio de la Biblia | 55 |
| Figura 29 Organización de actividades del piloto Experimental..... | 68 |
| Figura 30 Parte de la visualización de cronología de la Biblia (Aschmann, 2022) | 70 |
| Figura 31 Grafo del Dataset ideal (Grafica realizada con Código Python Anexo V)..... | 71 |
| Figura 32 Neo4j Sandbox para evaluar la herramienta de Base de datos de grafos | 73 |
| Figura 33 Ejemplo de una consulta con Cypher de Neo4j | 74 |
| Figura 34 Pantalla de desarrollo de Neo4j en el ejemplo creando a el libro de Nehemías | 75 |
| Figura 35 Otra vista ampliada de las relaciones del ejemplo la base de datos grafo Neo4j de libro de Nehemias | 75 |
| Figura 36 Vista de un ejemplo ampliado de los nodos y sus relaciones en el Libro de Nehemías con el personaje Esdras | 76 |
| Figura 37 Error 404. EL URI esta incorrecto o no existe | 78 |
| Figura 38 Dataset en GitHub con versiones de la biblia en texto | 79 |
| Figura 39 Pantallazo del sitio GitHub que contiene el código de Google Colab para leer una dataset texto..... | 79 |

Índice de Tablas

| | |
|--|-----------|
| Tabla 1 Simplificado Espacio vectorial de palabras | 24 |
| Tabla 2 Operación matemática vectorial en un espacio vectorial de palabras..... | 25 |
| Tabla 3 Etiquetas de la biblioteca spaCy de Posición y dependencia dentro de la oración | 39 |
| Tabla 4 Algunas etiquetas comunes de dependencia sintáctica | 45 |
| Tabla 5 Libros de la Biblia Hebrea o Antiguo testamento | 57 |
| Tabla 6 Libros Antiguo testamento mayoría biblias cristianas..... | 58 |
| Tabla 7 Antiguo testamento (Antiguo Testamento, n.d.)..... | 63 |
| Tabla 8 Libros del nuevo testamento más aceptados (Nuevo Testamento, n.d.) | 66 |

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 13.566.480 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 maneje 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 el estado del arte en el análisis de texto, las técnicas que se manejan en el procesamiento del lenguaje natural (NLP) y como el lenguaje de programación Python y la biblioteca spaCy trabajan revisando la cadena de producción o pipeline del análisis de texto para convertirse en una herramienta muy útil para la realización del análisis de texto de la biblia.

Luego se presentan los objetivos concretos.

Para el capítulo cuatro se presenta el desarrollo que se ha realizado en el prototipo y su desarrollo hasta el momento.

Para el capítulo cinco se tienen las conclusiones y el trabajo futuro que se propone de acuerdo a la experiencia.

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 NLP. (Vasiliev, 2020)

.

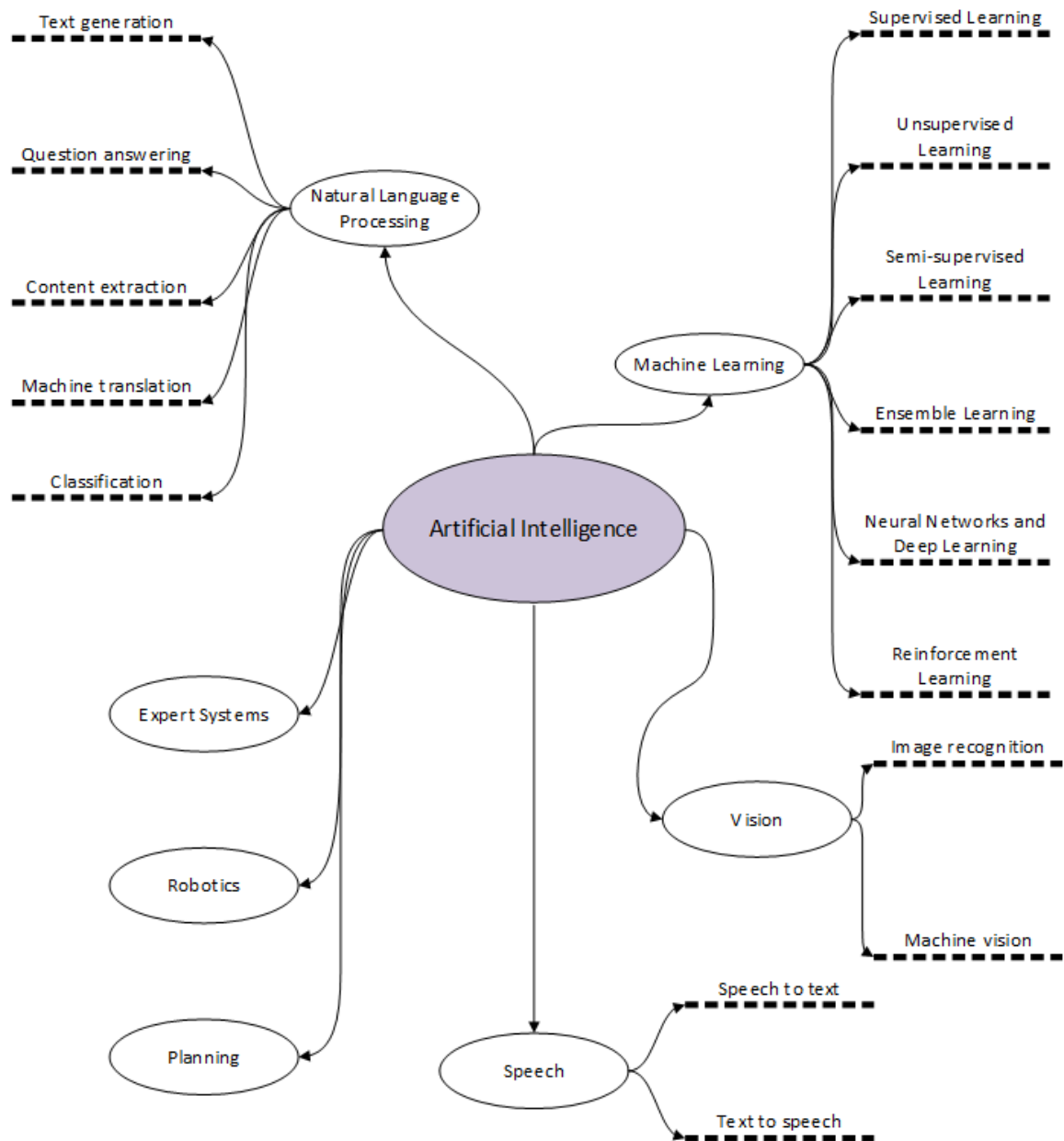


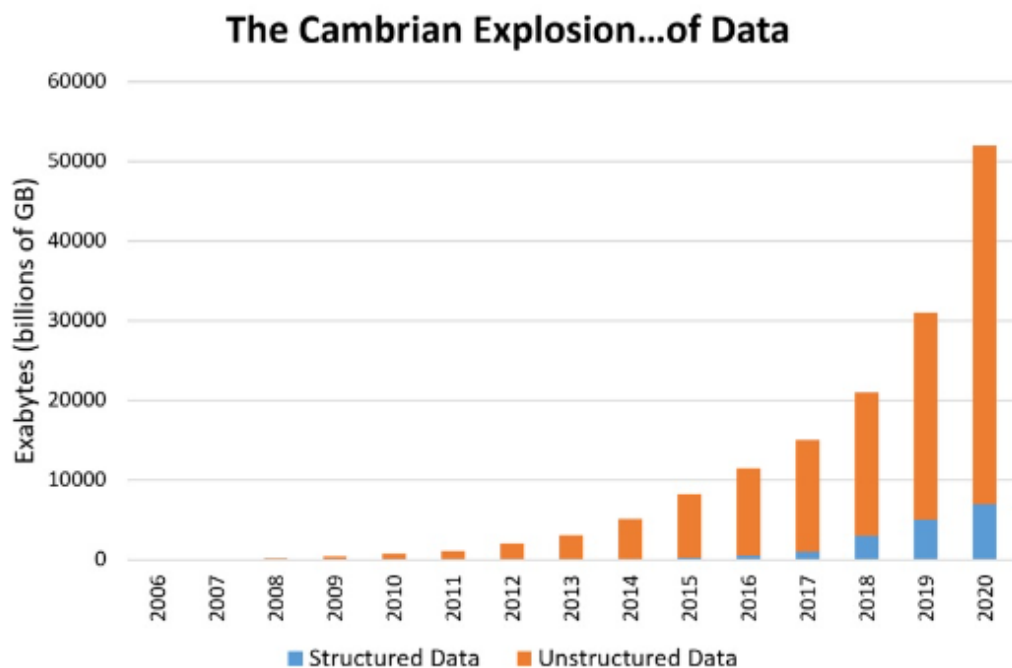
Figura 1 Taxonomía del machine Learning dentro de la Inteligencia Artificial. Basado en Panesar, 2019

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)



Source. Data originally from Patrick Cheesman's LinkedIn article.

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 hacer 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 de la oración (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.

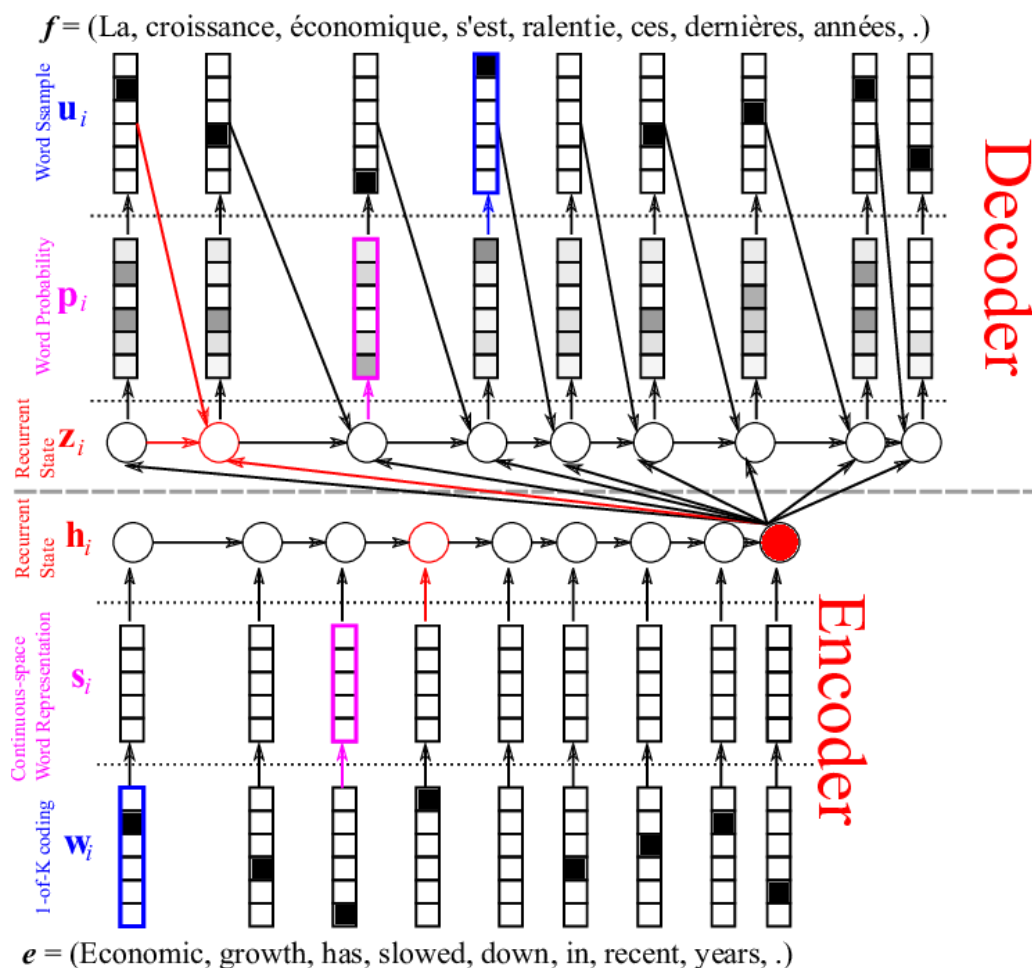


Figura 3 Un ejemplo de un modelo de traducción neuronal, trabajando del francés al inglés. (Introduction to Neural Machine Translation with GPUs (Part 2), 2015)

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.

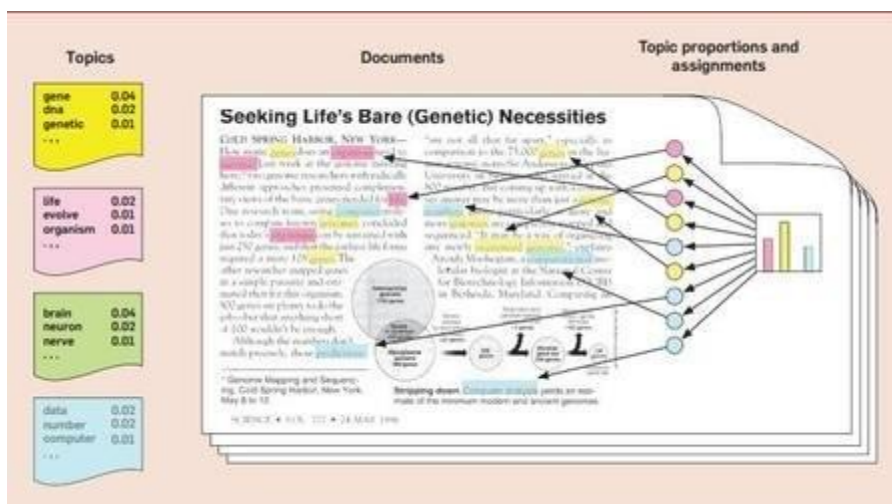


Figura 4 Técnicas como el modelado de temas utilizan métodos de modelado probabilístico para identificar temas clave del texto. (Thiyagarajan, 2018)

El experimentar la informática moderna en los teléfonos móviles, los recientes desarrollos tanto en Python como en NLP significan que ahora se puede desarrollar dichos sistemas por cuenta propia. No sólo ha habido una evolución en las técnicas utilizadas en NLP 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 NLP 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 textual de 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 de la oración, 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 NLP 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 Language 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 NLP, la recomendación de contenido y la visión por computadora. Los algoritmos comunes utilizados en diferentes contextos 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 (Convolutional 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 del ángulo 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 máquinas 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. Se 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.

| | País | 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.

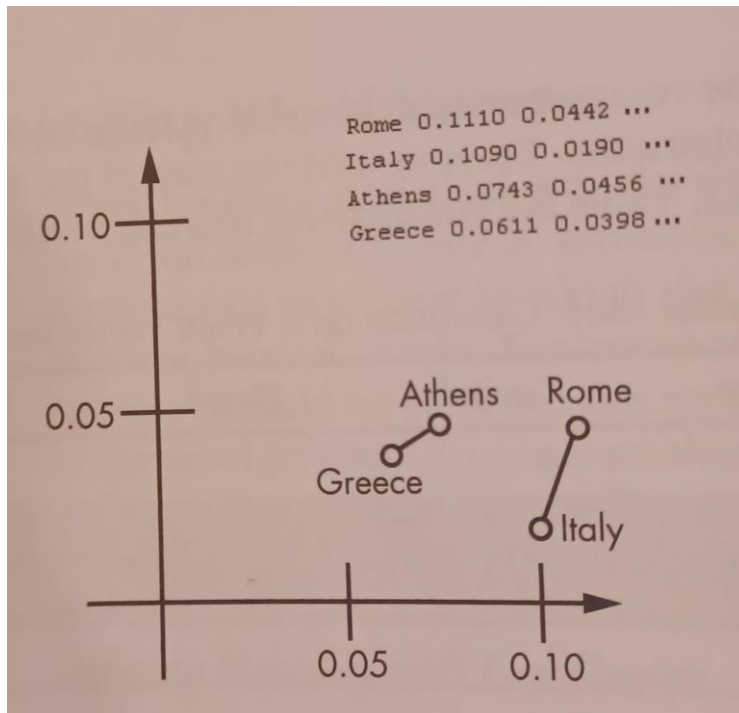


Figura 6 Un fragmento de una proyección de 2D

Una detalle interesante es que las líneas 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 biblioteca 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 calcular 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 span con un objeto documento y asi sucesivamente.

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

```
doc = nlp('Yo quiero una manzana verde')
```

```
doc.similarity(doc[2:5])
```

El resultado de la función de similitud es 0.7305813588233471

El código anterior es Python 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 comparando objetos consigo mismo el resultado es un 1. El método de similitud puede reconocer palabras que pertenecen a la misma o similar categoría y que frecuentemente aparecen relacionando contextos, mostrando un alto nivel de similitud escogiendo palabras claves para el cálculo de similitudes. El método de similitud calcula similitudes semánticas, 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 "Árboles 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 NLP 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.

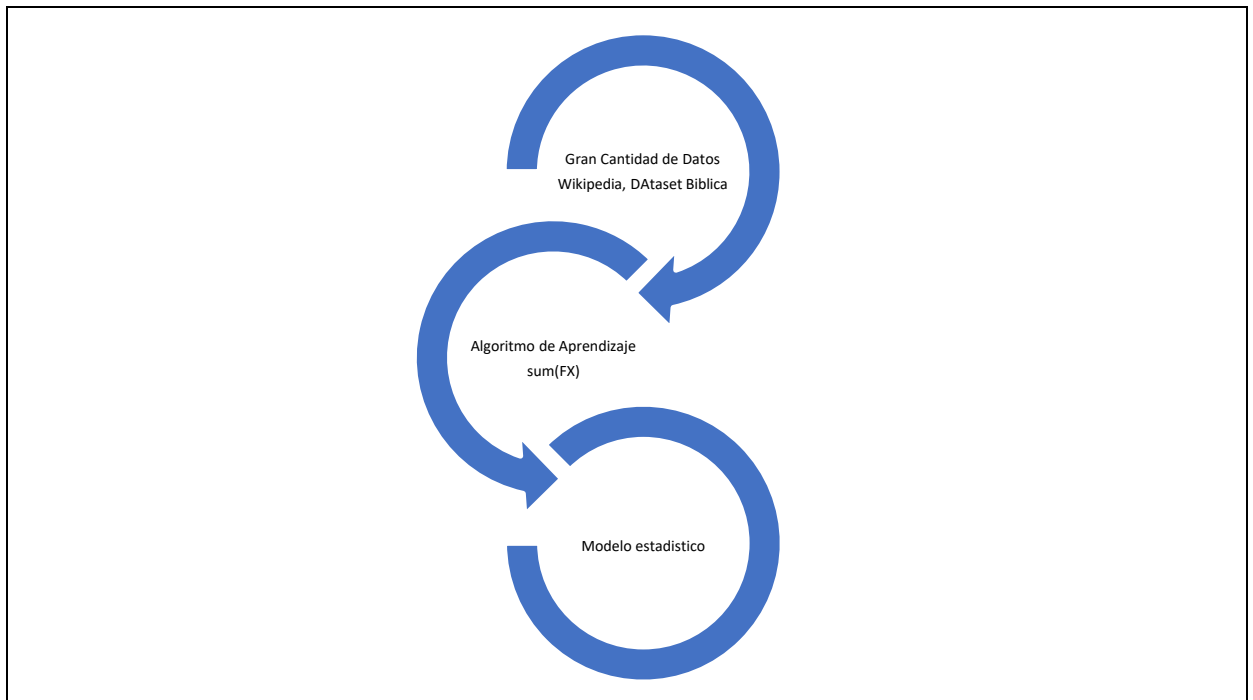


Figura 7 Generando un modelo estadístico con el algoritmo de aprendizaje automático

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 NLP. 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

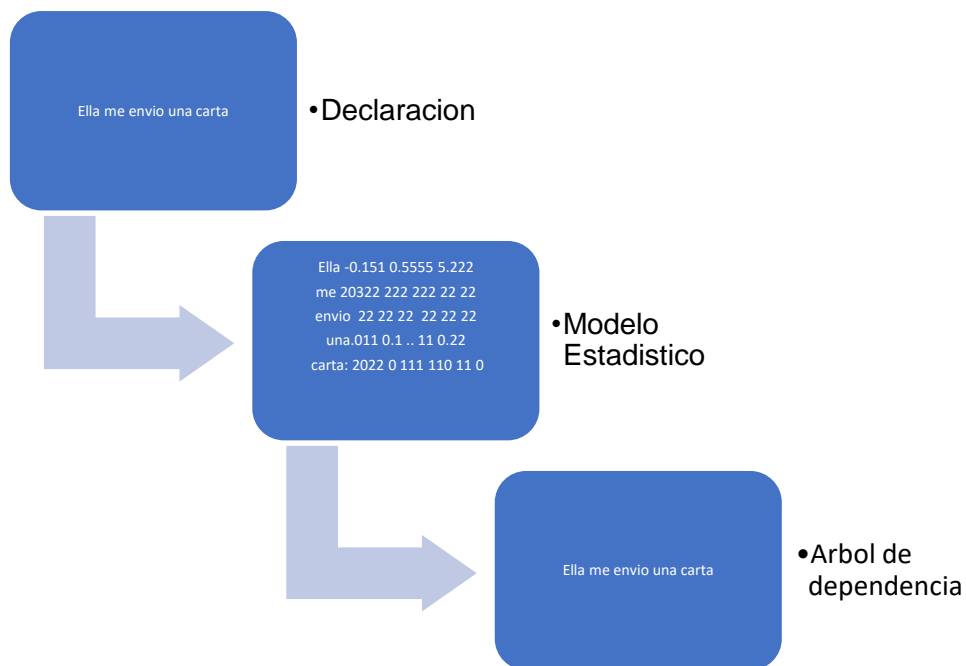


Figura 8 Ejemplo Predicción una estructura de árbol de dependencia usando un modelo estadístico

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.

2.2.5. 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, Gensim y scikit-learn que proporcionan herramientas poderosas para tareas como tokenización, 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 tensorflow 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 BeautifulSoup (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 tokenización, 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 embeddings) 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. (Transformers 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. (Vasilev, 2020)

La Figura siguiente proporciona una descripción simplificada de este proceso:

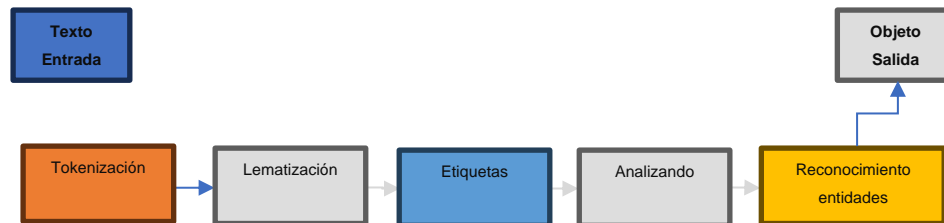


Figura 9 Vistazo General de los pasos de NLP

El proceso casi siempre incluye estos pasos Tokenización, lematización, etiquetado de parte del párrafo, análisis y reconocer los nombres de las entidades. (The central data structures in spaCy, n.d.)

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.

```
[1] !pip install spacy

[x] !python -m spacy download es_core_news_sm

[3] import spacy
    nlp = spacy.load('es_core_news_sm')
    doc = nlp(u'Estoy volando a Frisco')
    print([w.text for w in doc])

['Estoy', 'volando', 'a', 'Frisco']
```

Figura 10 Ejemplo de código Python haciendo Tokenización en el análisis de texto

El ejemplo se corre en la plataforma de Google Colab y el primer paso es cargar la librería spaCy, en el paso 2 cargar un 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:



```
# EJEMPLO DE LEMATIZACION
import spacy
nlp = spacy.load('es_core_news_sm')
doc = nlp(u'este producto integra ambas bibliotecas para descargar y aplicar mejoras')
for token in doc:
    print(token.text, token.lemma_)
```

este este
producto producto
integra integro
ambas ambos
bibliotecas biblioteca
para para
descargar descargar
y y
aplicar aplicar
mejoras mejora

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



| Token_Text | Token_Lema |
|-------------|------------|
| este | este |
| producto | producto |
| integra | integro |
| ambas | ambos |
| bibliotecas | biblioteca |
| para | para |
| descargar | descargar |
| y | y |
| aplicar | aplicar |
| mejoras | mejora |

Figura 12 Resultado del ejemplo Lematización

En la figura la columna de la izquierda contiene las fichas(tokens) 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 anterior:

Estoy volando a Frisco.

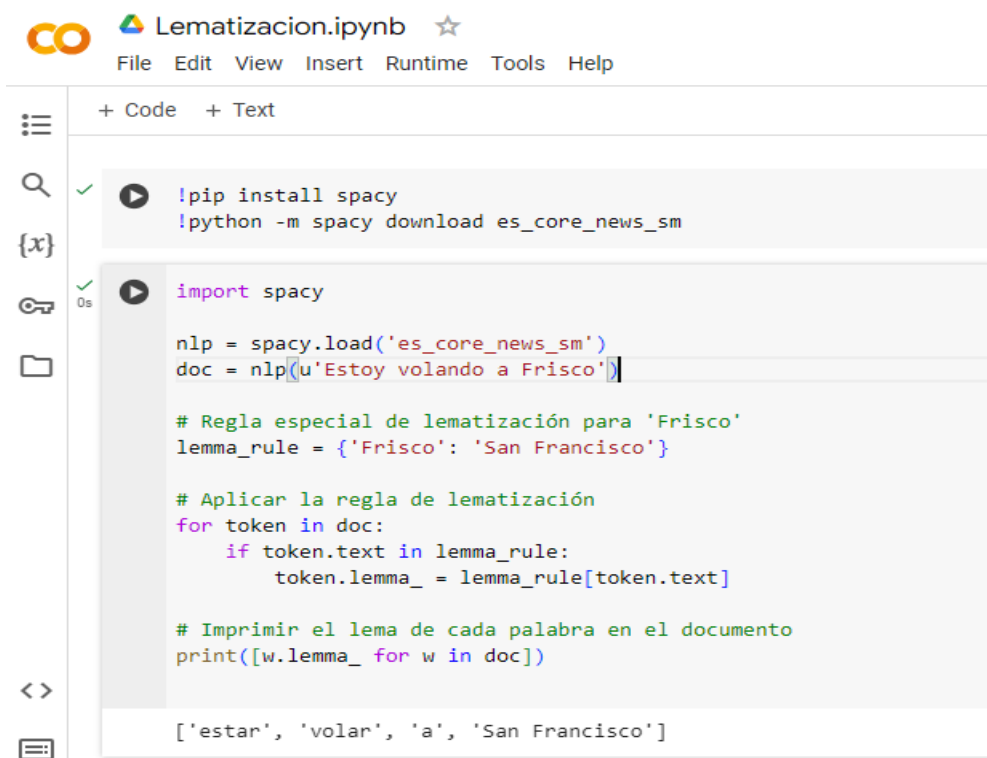
Supongamos que esta oración se envió a una aplicación de **NLP** que interactúa con un sistema en línea que proporciona una API para reservar boletos para viajes. La aplicación procesa la solicitud de un cliente, extrae de ella la información necesaria y luego la pasa a la API subyacente. Este diseño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



Figura 13 Usando Lematización en un requerimiento de un cliente con parte de la palabra

La aplicación NLP 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", "volando", "voló" y "volado" 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 apodosos 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 apodosos 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.



```
Lematizacion.ipynb
File Edit View Insert Runtime Tools Help

+ Code + Text

!pip install spacy
!python -m spacy download es_core_news_sm

import spacy

nlp = spacy.load('es_core_news_sm')
doc = nlp(u'Estoy volando a Frisco')

# Regla especial de lematización para 'Frisco'
lemma_rule = {'Frisco': 'San Francisco'}

# Aplicar la regla de lematización
for token in doc:
    if token.text in lemma_rule:
        token.lemma_ = lemma_rule[token.text]

# Imprimir el lema de cada palabra en el documento
print([w.lemma_ for w in doc])

['estar', 'volar', 'a', 'San Francisco']
```

Figura 14 Ejemplo Código Python usando biblioteca Spacy para lematización

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:

```
['Estoy', '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.

2.2.5.3.3 Etiqueta

Etiquetar una parte de la oración es decir que parte de la oración es la palabra o el texto analizado, ya sea un sustantivo, verbo, abjetivo, adverbio, conjunción y otros que son gramaticalmente determinados en la estructura de la oración. Una palabra puede actuar como otra parte de la oración, dependiendo del contexto,

En spaCy, las etiquetas de parte de la oración pueden incluir información detallada sobre un token. En el caso de los verbos, podrían indicarle las siguientes características: tiempo ya sea pasado, presente o futuro, o que aspecto ya sea simple, progresivo o perfecto, y en qué persona primera, segunda o tercera y si es singular o plural

Extraer estas etiquetas de partes de la oración 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 NLP 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 (LA). 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 considerar los lemas "volar" 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 de la oración. En español parte de la frase incluye sustantivo, pronombre, determinante, adjetivo, verbo, adverbio, preposición, conjunción e interjección.

Las marcas y otras se denominan partes de la oración 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 que son parte de la oración detalladas que brindan información más detallada sobre un token, cubriendo características morfológicas, como tiempos verbales y tipos de pronombres. Naturalmente, la lista de partes de la oración detalladas contiene muchas más etiquetas que la lista detallada. Las etiquetas de parte de la oración detalladas están disponibles como `Token.tag(int)` y atributos `token.tag_((Unicode))`.

| <u>Etiquetas POS</u> | <u>Etiquetas de Dependencia</u> |
|--|--|
| `NOUN`: Sustantivo | `nsubj`: Sujeto nominal |
| `VERB`: Verbo | `ROOT`: Raíz de la oración |
| `ADJ`: Adjetivo | `dobj`: Objeto directo |
| `ADV`: Adverbio | `nmod`: Modificador nominal |
| `PRON`: Pronombre | `amod`: Modificador adjetival |
| | `advmod`: Modificador adverbial |
| `DET`: Determinante | `prep`: Preposición |
| `ADP`: Adposición (preposición o postposición) | `conj`: Conjunción |
| `CONJ`: Conjunción | `det`: Determinante |
| `INTJ`: Interjección | `punct`: Puntuación |
| | |
| `NUM`: Número | |

Tabla 3 Etiquetas de la biblioteca spaCy de Posición y dependencia dentro de la oración

El tiempo y el aspecto son quizás las propiedades más interesantes de los verbos para aplicaciones de NLP. Juntos, indican la referencia de un verbo a una posición en el tiempo. Por ejemplo, la forma progresiva en tiempo presente de un verbo para describir lo que está sucediendo ahora o lo que sucederá, se agrega en tiempo presente del verbo "estar" y se adiciona el gerundio al verbo que expresa la acción. El gerundio se forma con la raíz del verbo añadiendo la terminación "-ando" para los verbos en -ar y -iendo para los verbos terminados en -er e -ir. Por ejemplo, el verbo orar en presente progresivo, al terminar en -ar se le añade -ando al verbo quedando "estoy orando".

Uso de etiquetas de parte de la oración para encontrar verbos relevantes

La aplicación de reserva de boletos podría usar las etiquetas detalladas de partes de la oración 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: "volar", "a" y "LA". Aquí hay algunas opciones simples:

Volé a Los Ángeles.

He volado a Los Ángeles.

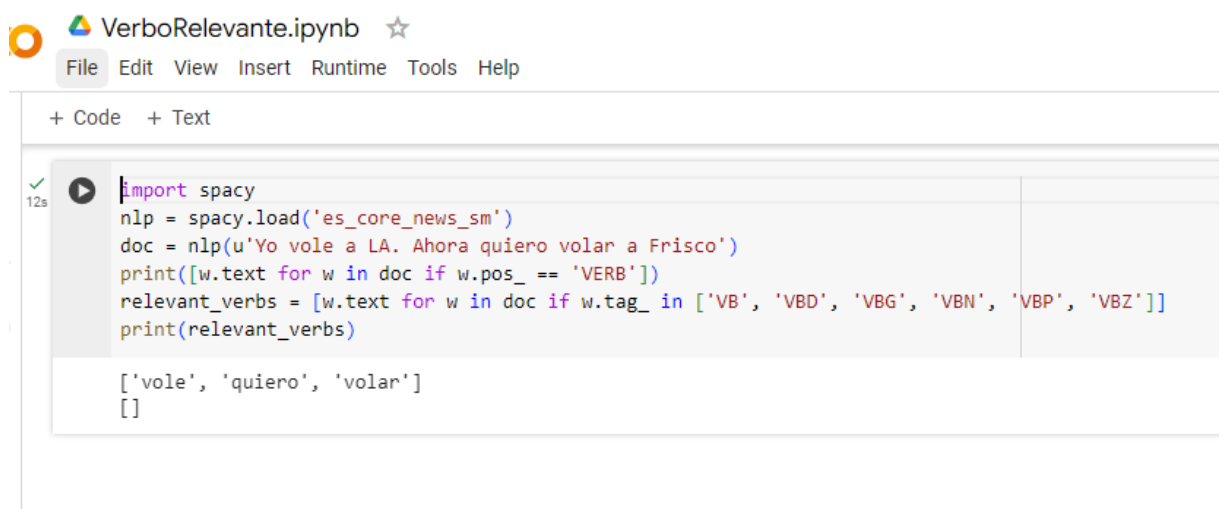
Necesito volar a Los Ángeles.

Estoy volando a Los Ángeles.

Volaré a Los Ángeles.

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 código en Python donde se generan los verbos relevantes usando las etiquetas de la oración.



```
import spacy
nlp = spacy.load('es_core_news_sm')
doc = nlp(u'Yo vole a LA. Ahora quiero volar a Frisco')
print([w.text for w in doc if w.pos_ == 'VERB'])
relevant_verbs = [w.text for w in doc if w.tag_ in ['VB', 'VBD', 'VBG', 'VBN', 'VBP', 'VBZ']]
print(relevant_verbs)

['vole', 'quiero', 'volar']
[]
```

Figura 15 Código que genera los verbos relevantes de la oración.

Las etiquetas detalladas de partes de la oración no solo se asignan a los verbos; también se asignan a las otras partes de la 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. El siguiente código muestra los verbos y los nombres propios. Como LA no lo encuentra como nombre propio se hace un ajuste.



```
[1] !pip install spacy

[2] !python -m spacy download es_core_news_sm

import spacy
nlp = spacy.load('es_core_news_sm')
doc = nlp(u'Yo vole a LA. Ahora quiero volar a Frisco')
print("Verbos:", [w.text for w in doc if w.pos_ == 'VERB'])

# Buscar nombres propios
proper_nouns = [w.text for w in doc if w.text.lower() == 'la' or w.pos_ in ['PROPN']]
print("Nombres propios:", proper_nouns)

Verbos: ['vole', 'quiero', 'volar']
Nombres propios: ['LA', 'Frisco']
```

Figura 16 Código para encontrar los verbos y los nombres propios.

El contexto es importante

Hay casos que las etiquetas detalladas de partes de la oración no siempre son suficientes para determinar el significado de una expresión, aquí el contexto es importante. Como ejemplo, la expresión: "Estoy volando a Los Ángeles". El etiquetador de parte de la oración asignará la etiqueta VBG al verbo "volar" en este ejemplo, porque está en la forma presente progresiva. Pero debido a que la forma verbal la oración puede significar "Ya estoy en el cielo, volando a Los Ángeles". o "Voy a volar a Los Ángeles". Cuando se envía a la aplicación NLP 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 la siguiente oración: "Voy a volar a Los Ángeles. Por la tarde tengo que regresar a Frisco". Lo más probable es

que esto implique que el usuario quiere un billete de avión de Los Ángeles a Frisco para un vuelo nocturno.

Relaciones sintácticas

En el análisis sintáctico de dependencias, cada palabra de una frase se representa individualmente como un "nodo" en una estructura de árbol conocida como árbol de análisis sintáctico. A continuación, los nodos se conectan mediante una relación de dependencia, que muestra la relación entre las palabras. Esto es útil para extraer la idea principal de una frase, así como para comprender cómo interactúan las palabras para construir frases.

Ahora si se combinan los nombres propios con el verbo que el etiquetador seleccionó anteriormente. La lista de verbos que podría utilizar para identificar la intención de la oración 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 compondría 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), se 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 de la oración las etiquetas de dependencia 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".

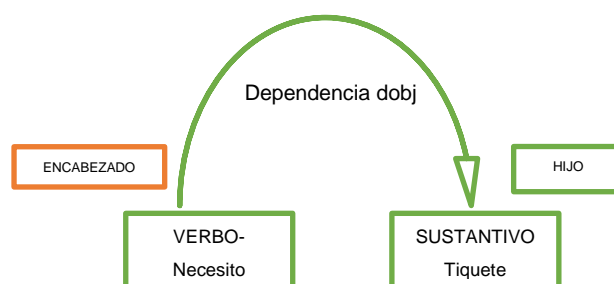


Figura 18 Ejemplo representación gráfica de la dependencia

CABEZA Y HIJO

Una etiqueta de dependencia sintáctica describe el tipo de relación sintáctica entre dos palabras en una oración. En tal par, una palabra es el gobernador sintáctico (también llamado cabeza o padre) y la otra es el dependiente (también llamado hijo). spaCy asigna una etiqueta de dependencia sintáctica al dependiente del par. Por ejemplo, en el par "necesitar, boleto", extraído de la oración "Necesito un boleto de avión", la palabra "boleto" es el niño y la palabra "necesitar" es la cabeza, porque "necesitar" es el verbo en lo que llamado frase verbal. En esta misma oración, "un billete de avión" es un sintagma nominal: el sustantivo "billete" es el principio, y "a" y "avión" son sus hijos.

Cada palabra de una oración tiene exactamente un encabezado. En consecuencia, una palabra sólo puede ser hijo de una cabeza. No siempre ocurre lo contrario. Una misma palabra puede actuar como cabeza en ninguno, uno o varios pares. Esto último significa que el jefe tiene varios hijos. Esto explica por qué siempre se asigna una etiqueta de dependencia al hijo.

La etiqueta dobj se asigna a la palabra "tiquete" porque es hijo de la relación. Siempre se asigna una etiqueta de dependencia al hijo. En el código se puede determinar el encabezado de una relación usando el atributo `Token.head`

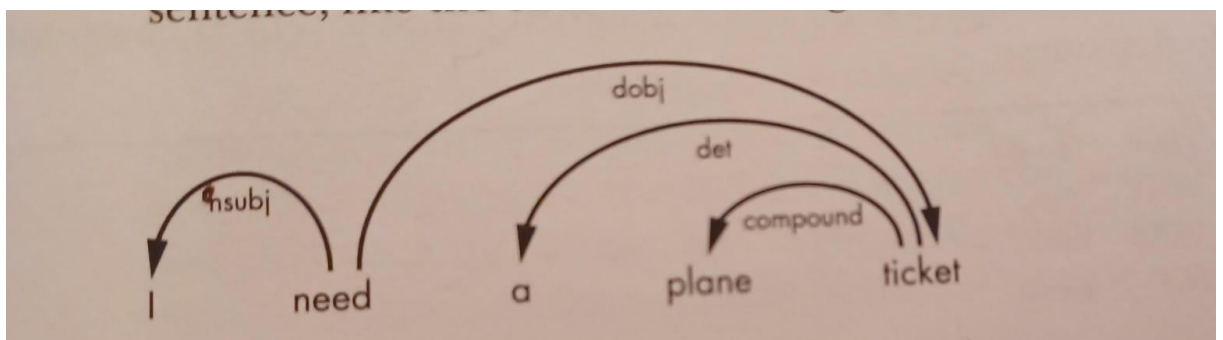


Figura 19 Relaciones de Encabezado/hijo en una oración completa

Como puedes ver en la figura 18, una misma palabra en una oración puede participar en varias relaciones sintácticas.

Algunas etiquetas de dependencia comunes

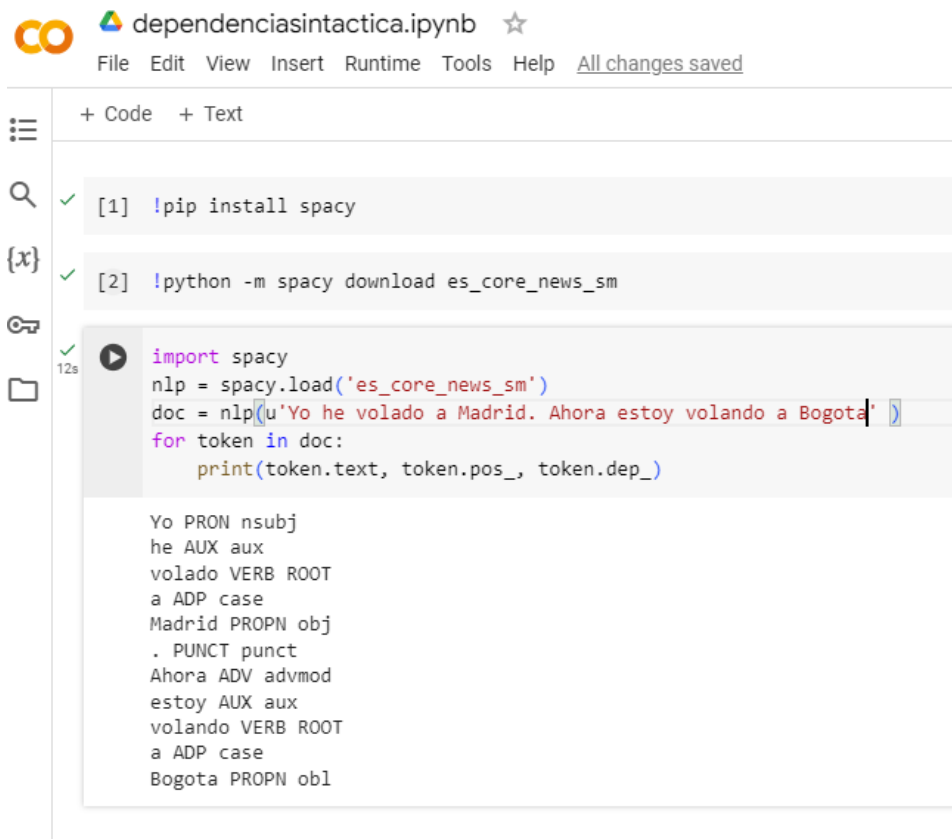
| Dependencia | Descripción |
|-------------|--|
| acomp | La dependencia "acomp" indica que la palabra a la que se le asigna esta dependencia es un complemento que acompaña al sustantivo o verbo principal en la oración. Este complemento puede ser un adjetivo u otra construcción que aporta información adicional sobre el sustantivo o verbo principal. |
| amod | La dependencia "amod" se utiliza para indicar que una palabra (generalmente un adjetivo) modifica a un sustantivo en la oración. Este modificador proporciona información adicional sobre las características o propiedades del sustantivo al que está vinculado. |
| aux | Los verbos auxiliares son aquellos que se utilizan junto con el verbo principal para formar construcciones verbales más complejas, como los tiempos verbales compuestos, la voz pasiva, las preguntas, entre otros. |
| compound | Esta dependencia se utiliza para indicar que una palabra compuesta está formada por dos o más palabras que trabajan juntas como una unidad semántica. |
| dative | Esta dependencia se utiliza para indicar la relación de un sustantivo o pronombre que funciona como un objeto indirecto en una oración. El objeto indirecto generalmente recibe la acción del verbo y suele expresar a quién o para quién se realiza la acción. En muchas oraciones, el objeto indirecto se encuentra después del verbo y antes del objeto directo. |
| det | Esta dependencia se utiliza para indicar la relación entre un determinante y el sustantivo al que modifica. Los determinantes son palabras que acompañan a los sustantivos y proporcionan información adicional sobre ellos, como la cantidad, la posesión o la identificación. Algunos ejemplos de determinantes en español son "el", "la", "un", "una", "mi", "su", etc. |
| dobj | Esta dependencia se utiliza para indicar la relación entre un verbo y su objeto directo, que es el sustantivo o pronombre que recibe directamente la acción del verbo. |
| nsubj | Esta dependencia se utiliza para indicar la relación entre un verbo y su sujeto, que es el sustantivo o pronombre que realiza la acción expresada por el verbo. |
| adp case | Esta relación específica indica la función gramatical de un sustantivo o pronombre que sirve como objeto de una |

| Dependencia | Descripción |
|-------------|--|
| | preposición. Cuando encuentras la combinación "adp case," se está señalando la preposición ("adp") y su objeto de caso preposicional. La etiqueta "case" se utiliza para indicar que el sustantivo o pronombre está en una relación de caso preposicional con la preposición. |
| obl | Esta dependencia se utiliza para indicar que un sustantivo o pronombre está actuando como objeto de una preposición, pero no está directamente gobernado por la preposición. Cuando se utiliza "obl", generalmente se está señalando que hay una construcción más compleja en la que el sustantivo u objeto está vinculado a la preposición de manera indirecta, posiblemente a través de otras palabras o construcciones. |
| advmod | Esta dependencia se utiliza para indicar que una palabra está actuando como un modificador de un adverbio en una oración. Un modificador adverbial afecta el significado del adverbio al que se adjunta. Puede proporcionar información adicional sobre la manera, el lugar, el tiempo, la frecuencia u otras circunstancias relacionadas con el verbo, adjetivo o adverbio al que modifica. |
| Pobj | Esta dependencia se utiliza para indicar la relación entre una preposición y el sustantivo o pronombre que sirve como objeto directo de esa preposición. |
| Root | La raíz es el nodo principal desde el cual se derivan todas las demás dependencias en la estructura del árbol sintáctico. En una oración, el verbo principal suele ser la raíz de la estructura sintáctica. La dependencia "root" indica la relación directa entre la raíz y el verbo principal que gobierna toda la oración. |

Tabla 4 Algunas etiquetas comunes de dependencia sintáctica

La etiqueta R00T marca el token cuyo encabezado 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.

El siguiente código de programación cómo acceder a las etiquetas de dependencia sintáctica de los tokens en el discurso del ejemplo de "Etiquetado de partes de la oración"



```
dependenciesintactica.ipynb ☆
File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

[1] !pip install spacy

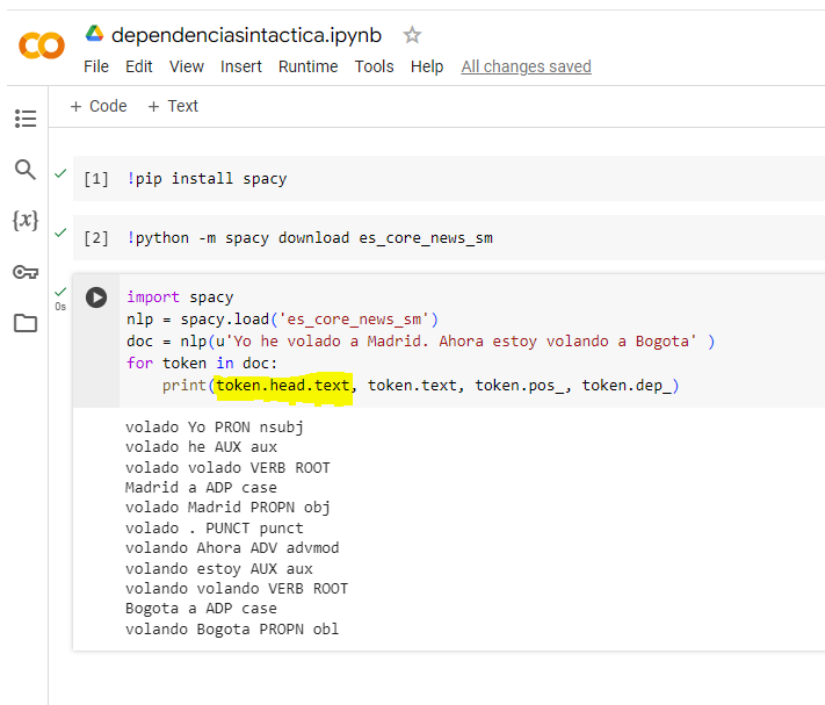
[2] !python -m spacy download es_core_news_sm

import spacy
nlp = spacy.load('es_core_news_sm')
doc = nlp(u'Yo he volado a Madrid. Ahora estoy volando a Bogota' )
for token in doc:
    print(token.text, token.pos_, token.dep_)

Yo PRON nsubj
he AUX aux
volado VERB ROOT
a ADP case
Madrid PROPN obj
. PUNCT punct
Ahora ADV advmod
estoy AUX aux
volando VERB ROOT
a ADP case
Bogota PROPN obl
```

Figura 20 Código para encontrar dependencia sintáctica

Para ver los arcos de dependencia en la oración solo es adicionar token.head.text,



```
dependenciesintactica.ipynb ☆
File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

[1] !pip install spacy

[2] !python -m spacy download es_core_news_sm

import spacy
nlp = spacy.load('es_core_news_sm')
doc = nlp(u'Yo he volado a Madrid. Ahora estoy volando a Bogota' )
for token in doc:
    print(token.head.text, token.text, token.pos_, token.dep_)

volado Yo PRON nsubj
volado he AUX aux
volado volado VERB ROOT
Madrid a ADP case
volado Madrid PROPN obj
volado . PUNCT punct
volando Ahora ADV advmod
volando estoy AUX aux
volando volando VERB ROOT
Bogota a ADP case
volando Bogota PROPN obl
```

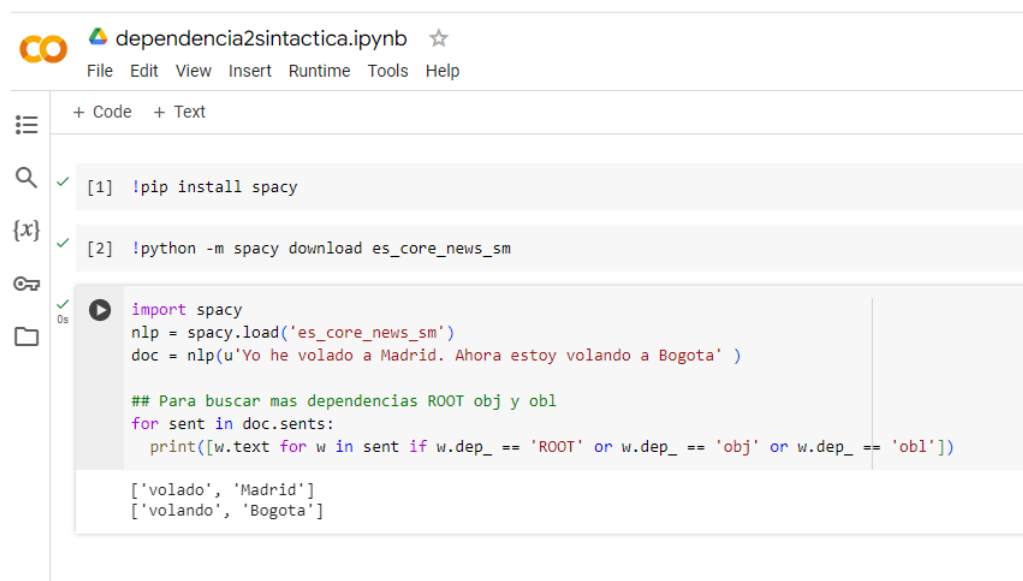
Figura 21 Código mostrando la raíz de la oración.

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 raíz de la relación. La razón es que la palabra a la que se asigna esta etiqueta es la única palabra en una oración que no tiene encabezado (figura 20).

Ahora para descubrir las etiquetas en los tokens que potencialmente describen mejor la intención del cliente se necesita encontrar un par que por sí solo describa apropiadamente la intención del cliente.

Para obtener tokens etiquetados con ROOT y dependencias etiquetadas de objetos de preposición (obl) y objetos directos (obj) que son clave para un mejor significado de todo el enunciado.

El siguiente código localiza las palabras asignadas a las dependencias root, obl y obj.



```
dependencia2sintactica.ipynb
File Edit View Insert Runtime Tools Help

+ Code + Text

[1] !pip install spacy

[2] !python -m spacy download es_core_news_sm

import spacy
nlp = spacy.load('es_core_news_sm')
doc = nlp(u'Yo he volado a Madrid. Ahora estoy volando a Bogota' )

## Para buscar mas dependencias ROOT obj y obl
for sent in doc.sents:
    print([w.text for w in sent if w.dep_ == 'ROOT' or w.dep_ == 'obj' or w.dep_ == 'obl'])

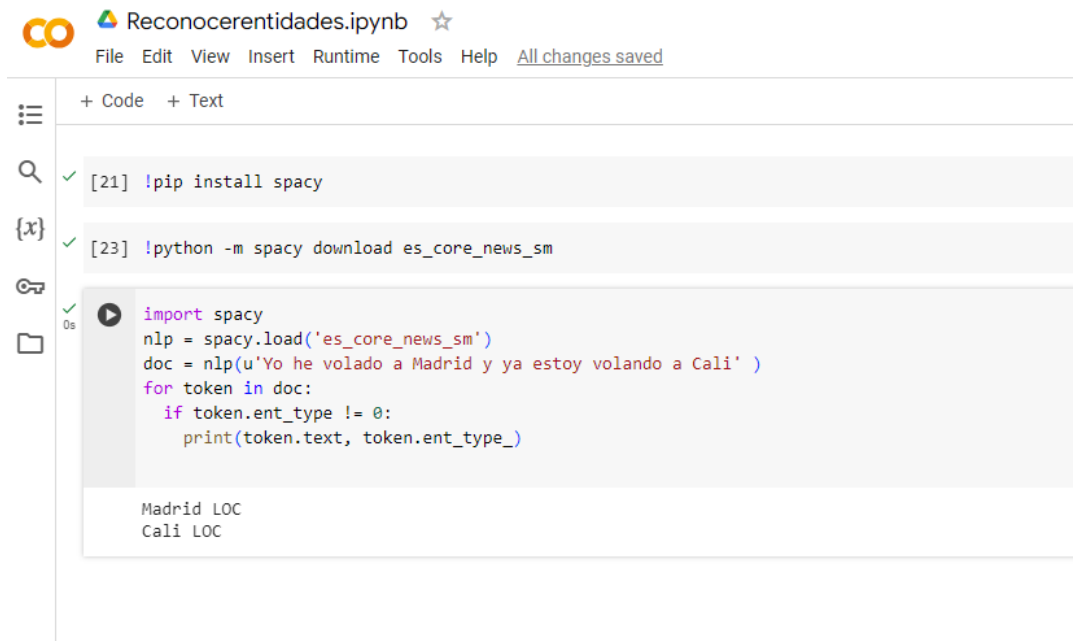
['volado', 'Madrid']
['volado', 'Bogota']
```

Figura 22 Código para identificar más dependencias y más entendimiento sintáctico.

En spaCy sents es una abreviatura de oraciones en inglés y se utiliza para dividir un documento en oraciones individuales. Cuando se procesa un texto con spaCy y se obtiene un objeto Doc, se puede acceder a las oraciones mediante el atributo sents. Cada elemento en sents es una oración del documento y así se puede trabajar las dependencias en cada oración.

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 NLP porque revelan el lugar u organización de la que habla el usuario. El siguiente código busca entidades con nombre en la oración y genera resultados de localizaciones geográficas.



```
[21] !pip install spacy

[23] !python -m spacy download es_core_news_sm

import spacy
nlp = spacy.load('es_core_news_sm')
doc = nlp(u'Yo he volado a Madrid y ya estoy volando a Cali' )
for token in doc:
    if token.ent_type != 0:
        print(token.text, token.ent_type_)

Madrid LOC
Cali LOC
```

Figura 23 Código reconocimiento de localidades geográficas

Tanto Madrid como Cali están etiquetados como GPE, el acrónimo de "entidad geopolítica" e incluye países, ciudades, estados y otros nombres de lugares.

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.

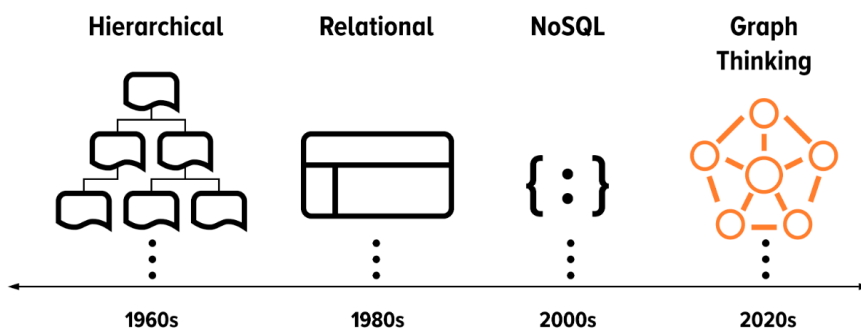
Todos los cuadernos de Colab se almacenan en Google Drive o puedes cargar desde GitHub. Los cuadernos de Colab se pueden compartir igual que los archivos de Documentos de Google.

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.

2.2.6. Dataset en base de Datos de Grafos (Neo4j)

Una breve historia de la evolución de la tecnología de las bases de datos se puede tomar desde 1960 hasta nuestros días. En la siguiente figura se puede ver que la última era es el pensamiento gráfico. (Broecheler, 2020)



Un dataset es esencialmente el conjunto de nodos, relaciones y propiedades que constituyen la información almacenada en la base de datos de grafos. La estructura de grafo proporciona una representación poderosa y flexible para modelar y analizar datos interconectados.

Existen varias bases de datos de grafos de código abierto que permiten almacenar y consultar datos en forma de grafos. Algunas de las más populares incluyen:

Neo4j es una de las bases de datos de grafos más conocidas y se utiliza ampliamente en la comunidad. Tiene una versión de código abierto llamada Community Edition y una versión comercial con características adicionales.

ArangoDB es una base de datos de múltiples modelos que admite documentos, grafos y claves-valor. Combina características de bases de datos de grafos, orientadas a documentos y clave-valor.

JanusGraph es una base de datos distribuida de grafos que se basa en Apache TinkerPop. Es escalable y puede manejar grandes conjuntos de datos distribuidos.

OrientDB es una base de datos de múltiples modelos que admite documentos, grafos y objetos. Puede ser utilizado tanto como base de datos de grafos como base de datos orientada a documentos.

Dgraph es una base de datos distribuida de grafos y orientada a documentos. Está diseñada para ser escalable y eficiente, y es particularmente adecuada para aplicaciones con grandes cantidades de datos interconectados.

Apache Giraph es un sistema de procesamiento de grafos basado en Apache Hadoop. Está diseñado para realizar operaciones de procesamiento de grafos a gran escala.

Con los requisitos específicos de piloto experimental se va a adentrar a revisar Neo4j.

Base de Datos de grafos Neo4j

En el contexto de Neo4j, una base de datos de grafos, el término "dataset" se puede entender como un conjunto de datos estructurados y organizados en forma de grafo. En Neo4j, un grafo es una estructura de datos que consiste en nodos, relaciones y propiedades, y se utiliza para representar y almacenar información de manera eficiente.

Un dataset en Neo4j se refiere a un conjunto de nodos, relaciones y propiedades que se almacenan en la base de datos de grafos. Un nodo representa una entidad, una relación describe la conexión entre dos nodos y las propiedades son atributos asociados a nodos o relaciones. Este conjunto de datos conforma la representación gráfica de la información y permite consultas y análisis eficientes basados en la estructura de grafos.

En Neo4j, los datasets se pueden utilizar para modelar y analizar **relaciones complejas** entre entidades. Por ejemplo, en el contexto bíblico un dataset podría incluir nodos que representan libros de la biblia, personajes, eventos y relaciones que indiquen algún vínculo, y propiedades que contienen información adicional como fechas o rangos de fechas de los eventos ocurridos. Esta representación gráfica facilita la navegación y consulta de patrones complejos de relaciones.

En la siguiente grafica hay se presenta un resumen de Neo4j:

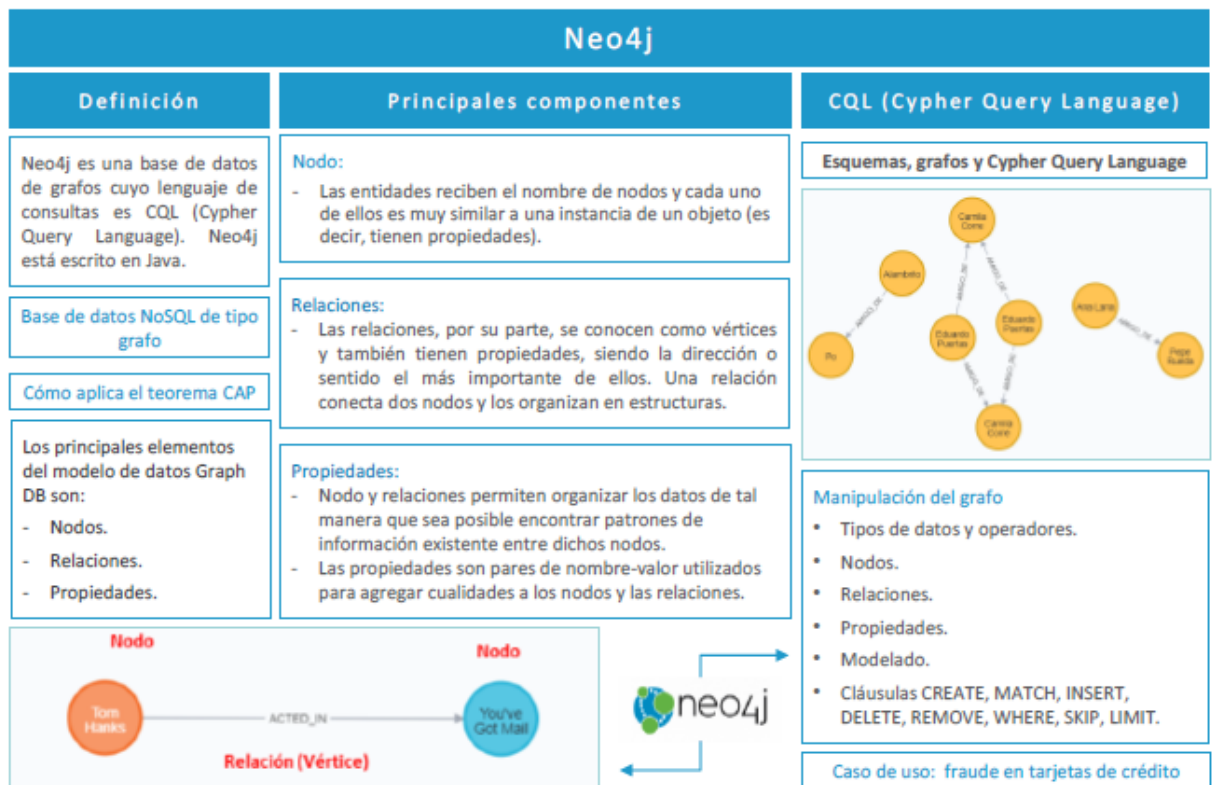


Figura 24 Esquema de Neo4j (Fuente: Tema 8 de clase Base de Datos para Big Data Universidad UNIR)

Los elementos de las bases de datos de grafo son:

Nodos: Los grafos representan entidades y las relaciones entre ellas. Las entidades reciben el nombre de todos y cada nodo es muy similar a una instancia de un objeto (es decir, tiene propiedades).

Relaciones: Las relaciones por su parte se conocen como vértices y también tienen propiedades, siendo la dirección o sentido el más importante de ellos. Una relación conecta dos nodos y los organiza en estructuras.

Propiedades: son pares de nombre-valor utilizados para agregar cualidades a los nodos y las relaciones.

De esta forma, un grafo puede mostrarse como una lista, un árbol, un mapa o una entidad compuesta, con la posibilidad de combinarse en estructuras aún más complejas y ricamente interconectadas (grandes y complejas redes de nodos).

Nodo y relaciones permiten organizar los datos de tal manera que sea posible encontrar patrones de información existente entre dichos nodos. Una vez almacenados los datos, los

grafos permitirán interpretar y generar información de valor de diferentes formas, según las relaciones que los nodos tengan.



Figura 25 Representación gráfica de nodo y de vértice

Recorrer los nodos en función de sus relaciones es lo que se conoce como **recorrido**.

El **esquema** hace referencia a los índices y restricciones que componen la base de datos.

Modelamiento de data grafica

Neo4j utiliza grafos para almacenar y administrar sus datos

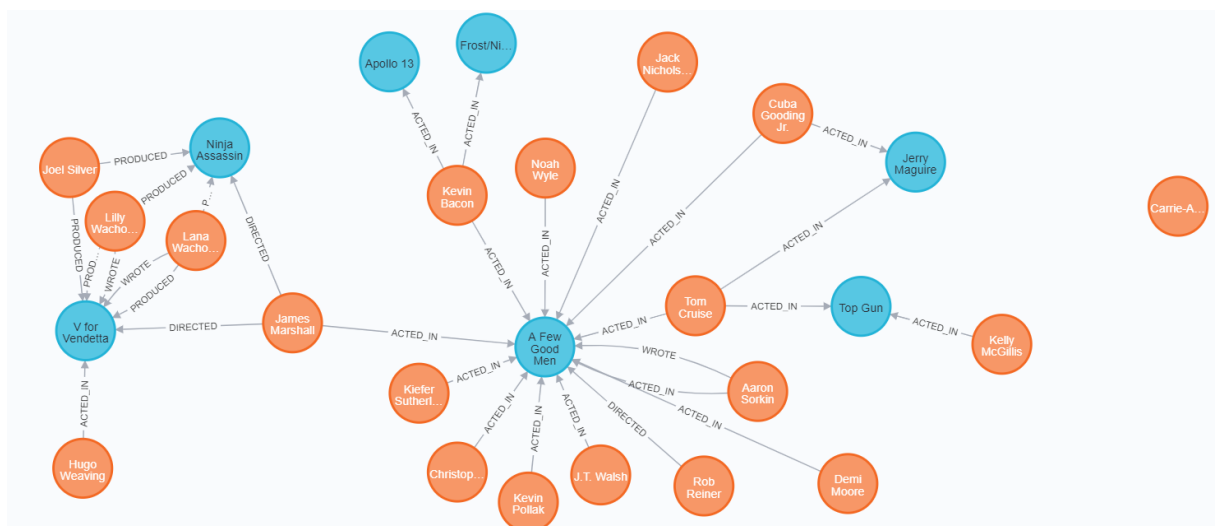


Figura 26 Representación de relaciones mediante grafos

El modelo de datos se representa con nodos (círculos), relaciones (flechas) y propiedades. Tanto los nodos como las relaciones contienen propiedades. Las propiedades también se representan como pares clave-valor. Las relaciones tienen direcciones unidireccionales y

bidireccionales. En una relación siempre hay un «nodo de inicio» o «nodo desde» y un «nodo hasta» o «nodo final». Una relación conecta dos nodos. Cuando se crean relaciones en Neo4j, siempre es necesario indicar una dirección, de lo contrario el motor de base de datos arrojará un mensaje de error.

Al almacenar los datos, no es necesario utilizar otra base de datos SQL o NoSQL para almacenar los datos de la base de datos de Neo4j. Toda la información se guarda en los distintos elementos en su formato nativo. Neo4j utiliza Native GPE (motor de procesamiento de grafos) para trabajar con su formato de almacenamiento de grafos nativo (Bernardino, 2018)

Para realizar consultas Neo4j utiliza un lenguaje llamado Cypher, que está diseñado específicamente para trabajar con datos de grafo y para la visualización hay herramientas que representa gráficamente las relaciones entre los datos, facilitando la comprensión.

Neo4j es también altamente escalable y puede manejar volúmenes de datos, flexible y adaptable a una variedad de casos de uso, que permite análisis complejos y sofisticados.

En la siguiente se muestra los componentes en una plataforma de grafos Neo4j.

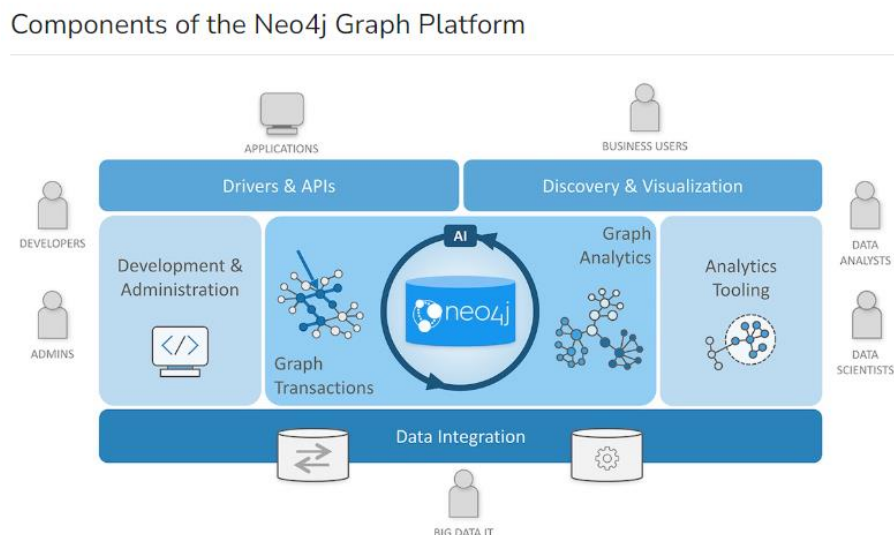


Figura 27 Componentes de Base de Grafos Neo4j

2.2.7. 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. Adicionalmente el estudio de la biblia plantea preguntas al mismo texto de quien la escribió, donde, cuando, para quien y por qué. Ente los objetivos se plantear que el mismo texto de la biblia sea que la que interprete y que se puedan encadenar textos paralelos, escritos por diferentes autores describiendo los mismos eventos.



Figura 28 Estudio de la Biblia

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

El antiguo testamento de los judíos escrito en hebreo antiguo, es el equivalente al antiguo testamento de la biblia de los cristianos. Se puede dividir en tres categorías: Tora, Profetas y escrituras

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

| Tora | Profetas | Escrituras |
|------|-----------|------------|
| | Sofonías | |
| | Hageo | |
| | Zacarías | |
| | Malaquías | |

Tabla 5 Libros de la Biblia Hebrea o Antiguo testamento

El antiguo testamento de la biblia para la mayoría de los cristianos está conformado por 39 libros y se clasifican en cuatro grupos de acuerdo a la misma clase de escritura:

| Grupo | Libros mayoría iglesias cristianas |
|------------|------------------------------------|
| Ley | Génesis |
| | Éxodo |
| | Levítico |
| | Números |
| | Deuteronomio |
| Históricos | Josué |
| | Jueces |
| | Ruth |
| | Samuel (I) |
| | Samuel (II) |
| | Reyes (I) |
| | Reyes (II) |
| | Crónicas (I) |
| | Crónicas (II) |
| | Esdras |
| | Nehemías |
| | Ester |
| Poesía | Job |
| | Salmos |
| | Proverbios |

| Grupo | Libros mayoría iglesias cristianas |
|----------|------------------------------------|
| Profetas | Eclesiastés |
| | Cantares |
| | Isaías |
| | Jeremías |
| | Lamentaciones |
| | Ezequiel |
| | Daniel |
| | Oseas |
| | Joel |
| | Amos |
| | Abdías |
| | Jonás |
| | Miqueas |
| | Nahum |
| | Habacuc |
| | Sofonías |
| | Hageo |
| | Zacarías |
| | Malaquías |

Tabla 6 Libros Antiguo testamento mayoría biblias cristianas

| <i>Tanaj</i> (Biblia judía) (24 libros) Los libros en negrita forman parte del <i>Ketuvim</i> | Antiguo Testamento Protestante (39 libros) | Antiguo Testamento Iglesia católica (46 libros) | Antiguo Testamento Iglesia ortodoxa (51 libros) | Idioma original |
|--|---|--|--|----------------------------------|
| <i>Torah</i> | <i>Pentateuco o los Cinco Libros de Moisés</i> | | | |

| <i>Tanaj</i> (Biblia judía) (24 libros) Los libros en negrita forman parte del <i>Ketuvim</i> | Antiguo Testamento Protestante (39 libros) | Antiguo Testamento Iglesia católica (46 libros) | Antiguo Testamento Iglesia ortodoxa (51 libros) | Idioma original |
|--|---|--|--|----------------------------------|
| 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 |
| <i>Nevi'im (Profetas)</i> | <i>Libros históricos</i> | | | |
| 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 |

| <i>Tanaj</i> (Biblia judía) (24 libros) Los libros en negrita forman parte del <i>Ketuvim</i> | Antiguo Testamento Protestante (39 libros) | Antiguo Testamento Iglesia católica (46 libros) | Antiguo Testamento Iglesia ortodoxa (51 libros) | Idioma original |
|--|---|--|--|----------------------------|
| 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 (Tobías) | Tobit (Tobías) | Arameo (¿y Hebreo?) |
| | | Judith | Judith | Hebreo |
| Esther⁵ | Esther | Esther | Esther | Hebreo |
| | | 1 Macabeos | I Macabeos | Hebreo |
| | | 2 Macabeos | II Macabeos | Griego |
| | | | III Macabeos | Griego |
| | | | IV Macabeos | Griego |

| <i>Tanaj</i> (Biblia judía) (24 libros) Los libros en negrita forman parte del <i>Ketuvim</i> | Antiguo Testamento Protestante (39 libros) | Antiguo Testamento Iglesia católica (46 libros) | Antiguo Testamento Iglesia ortodoxa (51 libros) | Idioma original |
|--|---|--|--|----------------------------|
| <i>Ketuvim (Escritos)</i> | <i>Libros sapienciales</i> | | | |
| 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 |
| <i>Nevi'im (Últimos Profetas)</i> | <i>Profetas mayores</i> | | | |
| Yeshayahu | Isaías | Isaías | Isaías | Hebreo |

| <i>Tanaj</i> (Biblia judía) (24 libros) Los libros en negrita forman parte del <i>Ketuvim</i> | Antiguo Testamento Protestante (39 libros) | Antiguo Testamento Iglesia católica (46 libros) | Antiguo Testamento Iglesia ortodoxa (51 libros) | Idioma original |
|--|---|--|--|------------------------------------|
| 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 |
| Daniel⁵ | Daniel | Daniel | Daniel | Hebreo y Arameo |
| | <i>Profetas menores</i> | | | |
| Los Doce o <i>Trei Asar</i> | Oseas | Oseas | Oseas | Hebreo |
| | Joel | Joel | Joel | Hebreo |
| | Amós | Amós | Amós | Hebreo |
| | Abdías | Abdías | Abdías | Hebreo |
| | Jonás | Jonás | Jonás | Hebreo |

| <i>Tanaj</i> (Biblia judía) (24 libros) Los libros en negrita forman parte del <i>Ketuvim</i> | Antiguo Testamento Protestante (39 libros) | Antiguo Testamento Iglesia católica (46 libros) | Antiguo Testamento Iglesia ortodoxa (51 libros) | Idioma original |
|--|---|--|--|----------------------------------|
| | 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 7 Antiguo testamento (Antiguo Testamento, n.d.)

Otro punto que de los autores hay controversias pero que en los mismos escritos se pueden extraer 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. (Jesús 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 (cerca de 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, los Hechos de los Apóstoles, las epístolas de Pablo de Tarso, siete epístolas católicas de diversa atribución y el Apocalipsis, como se puede observar en el esquema que se encuentra a continuación.

Comprende, en total, 27 libros en el canon 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 |

| Orden | Libro | Abreviatura | N.º Capítulos |
|-------|---------------------------------------|-------------|---------------|
| 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 |

| Orden | Libro | Abreviatura | N.º Capítulos |
|-------|---------------------------|-------------|---------------|
| 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 8 Libros del nuevo testamento más aceptados (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 de '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.

2.3 Conclusiones

La unión de las técnicas de análisis de texto usando herramientas de inteligencia artificial y el poder integrar las técnicas de estudio de los líderes, maestros y pastores con dataset bíblicos que estén completas y bien estructuradas pueden generar herramientas de visualización y consulta muy útiles para el análisis de las relaciones de los personajes bíblicos en los diferentes eventos a través del tiempo que data el texto bíblico.

Se está trabajando un grupo de pastores en Colombia (Anexo III) 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

4.1.1 Organización

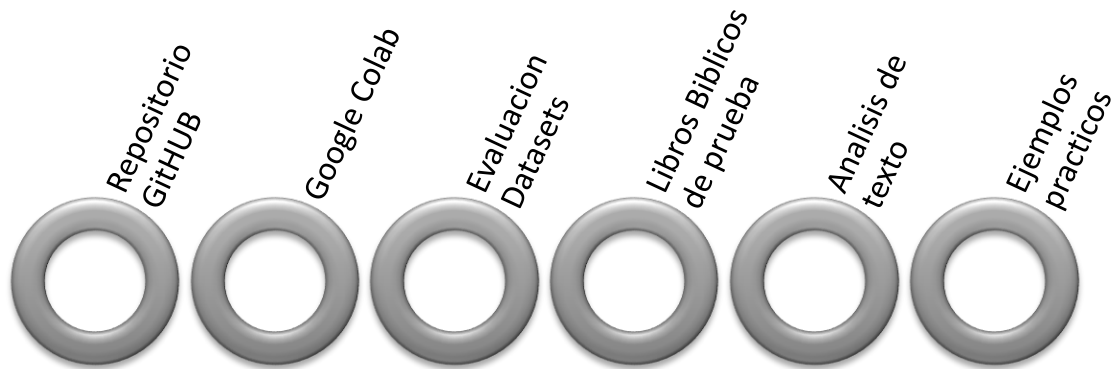


Figura 29 Organización de actividades del piloto Experimental

4.1.1.1. Repositorio

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.

4.1.1.2. Accesibilidad y uso de la nube.

Con Google Colab se ofrece el entorno de desarrollo basado en las herramientas de Jupyter 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.

Otra ventaja es que se obtiene **reproducibilidad del proyecto** conectándose con GitHub que gestiona las dependencias del proyecto. En los archivos `requirements.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 personas 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.

4.1.1.3 Evaluación Datasets

Con las datasets analizados en sitios públicos que son de fuente abierta (open source), se encuentran algunos proyectos que iniciaron y quedaron o están sin actualizarse en años y aún no están completos. Las estructuras y tecnologías de los datasets son diferentes en cada caso, lo que conlleva a realizar transformaciones y adecuaciones para utilizarlos en el prototipo. En varios casos se hace necesario obtener autorizaciones y no están fácilmente accesibles para desarrollar el prototipo.

Se buscaron datasets públicos de la biblia en plataformas (Anexo IV) como Kaggle, BiblIA, Data.gov, Nasa, GitHub donde hay una cantidad importante. 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 (Christodouloupoulos, n.d.). Para realizar un análisis de la cronología y relaciones entre personajes bíblicos y eventos a lo largo del tiempo, puede ser útil utilizar una combinación de fuentes de datos.

Un sitio web interesante dedicado a la cronología bíblica con autoría de Rick Aschman contiene un estudio que presenta los datos con una visualización que nos da una buena idea de la orientación de los datos para presentar eventos, personajes y fechas con una visualización en una línea de tiempo. (Aschmann, 2022)

Cronología de la Biblia

(cronologiabiblica.net) © 2022 Richard P. Aschmann

Cuadro actualizado 11-jun.-2022.

¿a esta cronología con Taré y no con Adán? Esto depende de la cuestión de si las 12 genealogías en Génesis 5 y Génesis 11? ¿No están completas? No, en realidad un alto porcentaje de las Escrituras nos obliga a concluir que no lo son. Lea más sobre esto [aquí](#). Si hay una parte de la cronología de la Biblia que puede establecerse antes de Taré, la rama del [Diluvio de Noé](#).

de caso, tanto un Creacionista de la Tierra Joven (CTJ) como un Creacionista de la CTA) debe poder aceptar la cronología a través de todo este cuadro, ya que se basa en que son independientes de esta cuestión.

Todos los nombres bíblicos en este cuadro siguen su forma en la Reina-Valera 1960. Algunas otras versiones, como la Nueva Versión Internacional, tienen formas de estos nombres que realmente son más similares a su forma original hebrea o griega, como Najor y Jarán en vez de Nacor y Harán, pero los nombres en la Reina-Valera son los tradicionales y los más conocidos.

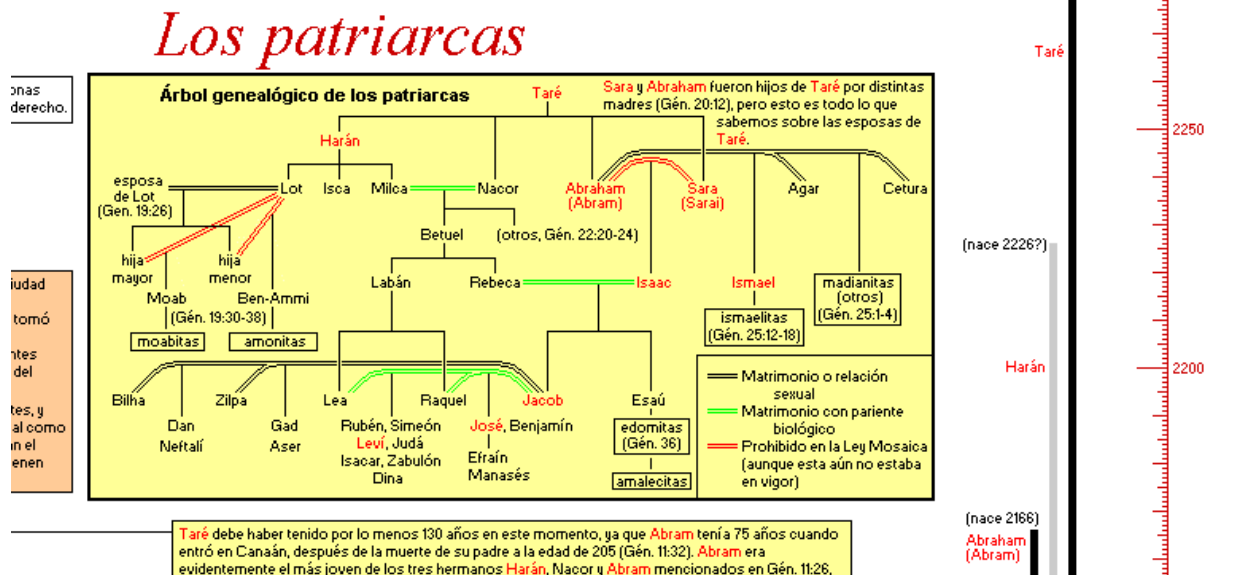


Figura 30 Parte de la visualización de cronología de la Biblia (Aschmann, 2022)

4.1.1.4. Dataset Ideal

El dataset ideal tiene que incluir textos bíblicos ya sea completos o parciales de la biblia en diferentes versiones y que tenga una estructura que permita versículos específicos relacionados con los eventos, personas y sus relaciones. Debe contener genealogías presentes en la biblia para seguir las líneas familiares y relaciones entre personajes, manejar diferentes calendarios y sus conversiones. También identificar eventos claves mencionados en la biblia, como éxodos, guerras, reinos, profecías cumplidas asociando fechas específicas o rangos en los cuales estos eventos ocurrieron. Otro dato interesante en lo ideal es incorporar datos geográficos para visualizar la ubicación de los eventos y lugares bíblicos en un mapa asociándolos con los lugares mencionados en la biblia. Datos que asocien el contexto histórico

y cultural de la época en que se desarrolla cada evento incluyendo información sobre imperios, costumbres sistemas políticos y sociales de la antigüedad. Referencias cruzadas entre personajes y eventos y que se pueda rastrear la continuidad de historias y personajes a lo largo de los libros de la biblia. Utilizar datos históricos y arqueológicos externos para complementar la cronología bíblica teniendo la posibilidad de combinar eventos bíblicos con eventos históricos documentados en otros textos externos a la biblia. Que contenga metadatos de los libros como autor fechas estimadas de escritura y cualquier otro detalle que sea importante. Información de las versiones o traducciones de la biblia. Que sea compatible con herramientas que faciliten el análisis del texto, su extracción y la visualización de datos.

La dataset requiere de un grupo o algún método formal que pueda dar aprobación para que la información tenga calidad y sea confiable teniendo en cuenta que la interpretación de textos bíblicos puede variar por lo que es muy útil incorporar al proyecto fuentes académicas y teológicas en cuanto a este punto y poder utilizar herramientas en el procesamiento de texto que utilizando IA pueda generar información útil de análisis y visualización.

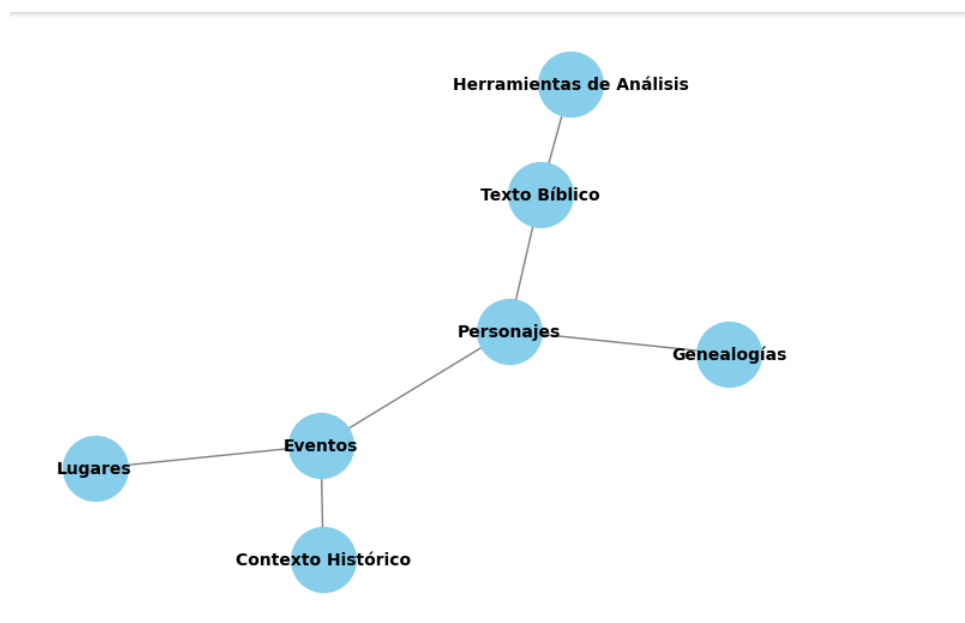


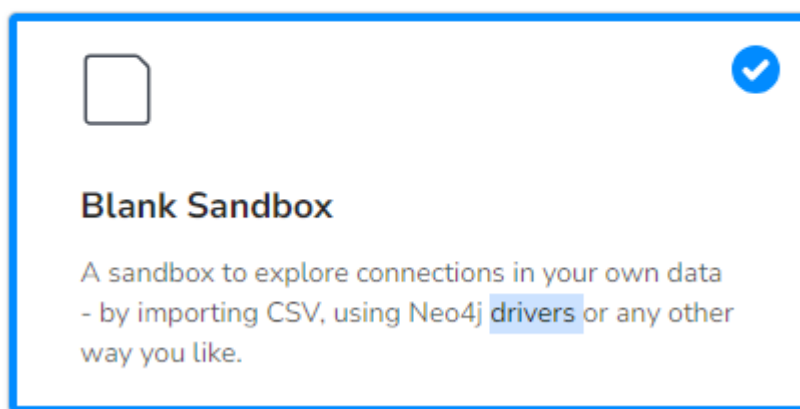
Figura 31 Grafo del Dataset ideal (Grafica realizada con Código Python Anexo V)

En la búsqueda de una base de datos que maneje en forma eficiente la cantidad de relaciones complejas de los campos definidos de acuerdo al dataset ideal se revisan herramientas tanto de bases de datos relacionales, como bases de datos no relacionales (NoSQL) que aporten al propósito de la investigación (Knowledge Base of Relational and NoSQL Database Management Systems , n.d.). La que mejor se ajusta es **Neo4j** que utiliza bases de datos de

grafos con herramientas y bibliotecas en lenguajes de programación como NetworkX en Python o igraph.

Para el proceso de evaluación se utiliza Neo4j Sandbox que da un ambiente de prueba desde el propio sitio web de Neo4j y se utiliza idealmente para aprendizaje, experimentación y pruebas de concepto, ya que proporciona entornos configurados sin necesidad de configurar una infraestructura propia. La limitación es que solo aplica para un tiempo de tres días, por lo que no es ideal para la aplicación en producción o un análisis que requiera más tiempo.

Your own data



Pre-built data



Project : **Blank Sandbox**



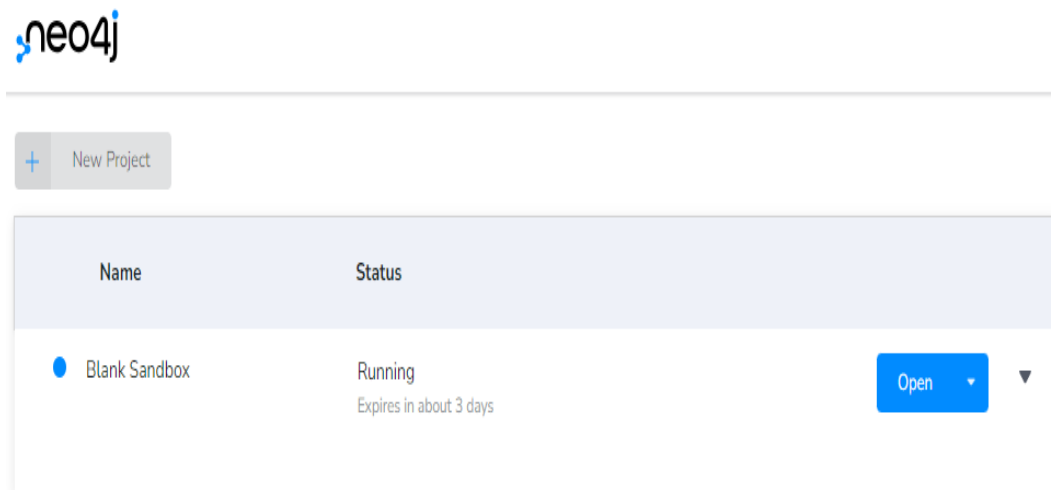
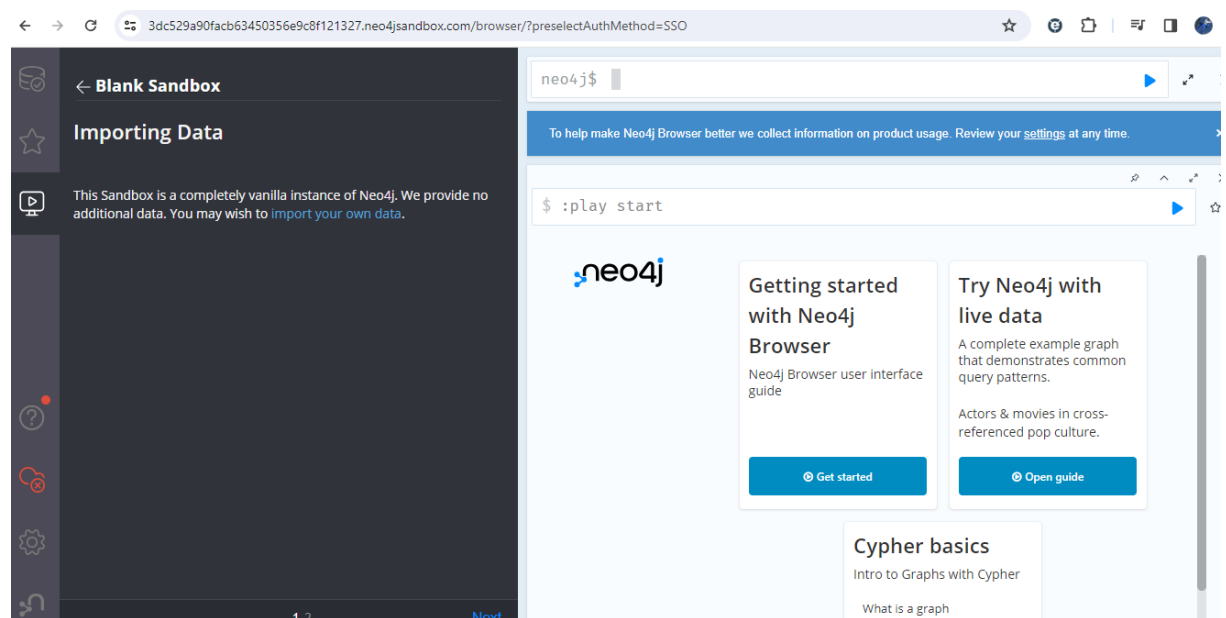


Figura 32 Neo4j Sandbox para evaluar la herramienta de Base de datos de grafos



Se crearon nodos, relaciones y propiedades y se hacen consultas utilizando Cypher y dan resultados como la siguiente grafica.

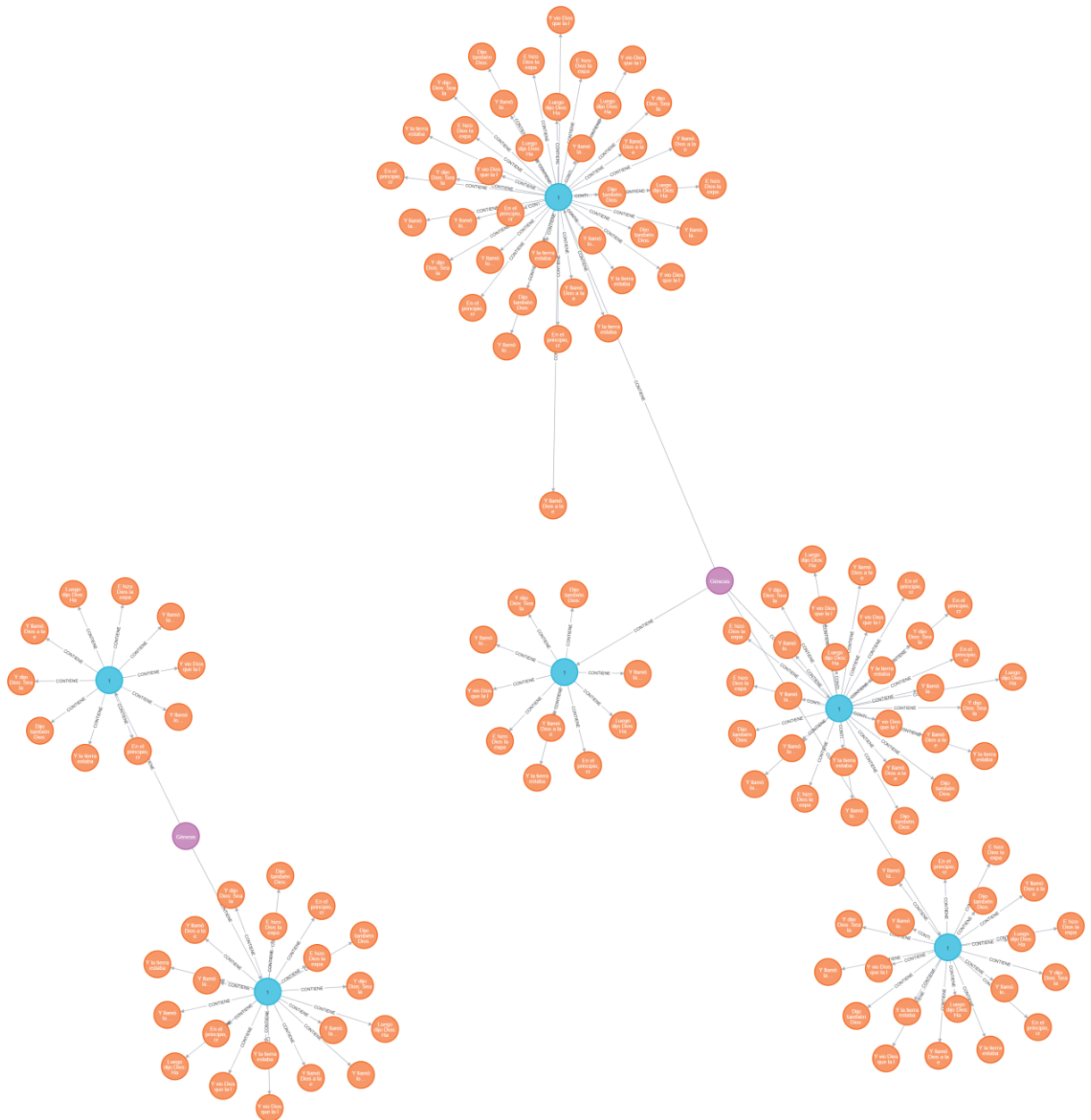


Figura 33 Ejemplo de una consulta con Cypher de Neo4j

Una ventaja en las consultas de Neo4j usando Cypher es su posibilidad a exportar a formato JSON dando la oportunidad de que sean dirigidas a un tema específico y de allí pasarla a un ambiente colaborativo como GitHub y así realizar algún análisis de texto que se requiera.

Por ejemplo, para crear el libro de Nehemías en Neo4j con sus capítulos, versículos, personajes, eventos y lugares utilizando la base de datos grafica usando un script en Cypher (Anexo VII)

En la gráfica se muestra el grafo de la base de datos en Neo4j Sandbox

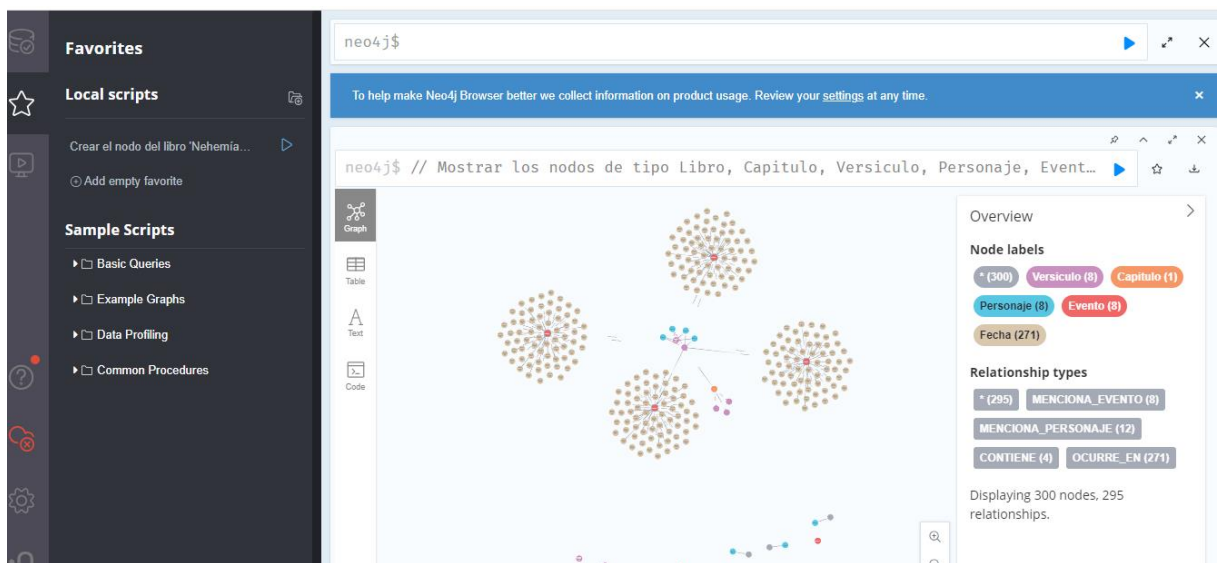


Figura 34 Pantalla de desarrollo de Neo4j en el ejemplo creando a el libro de Nehemías

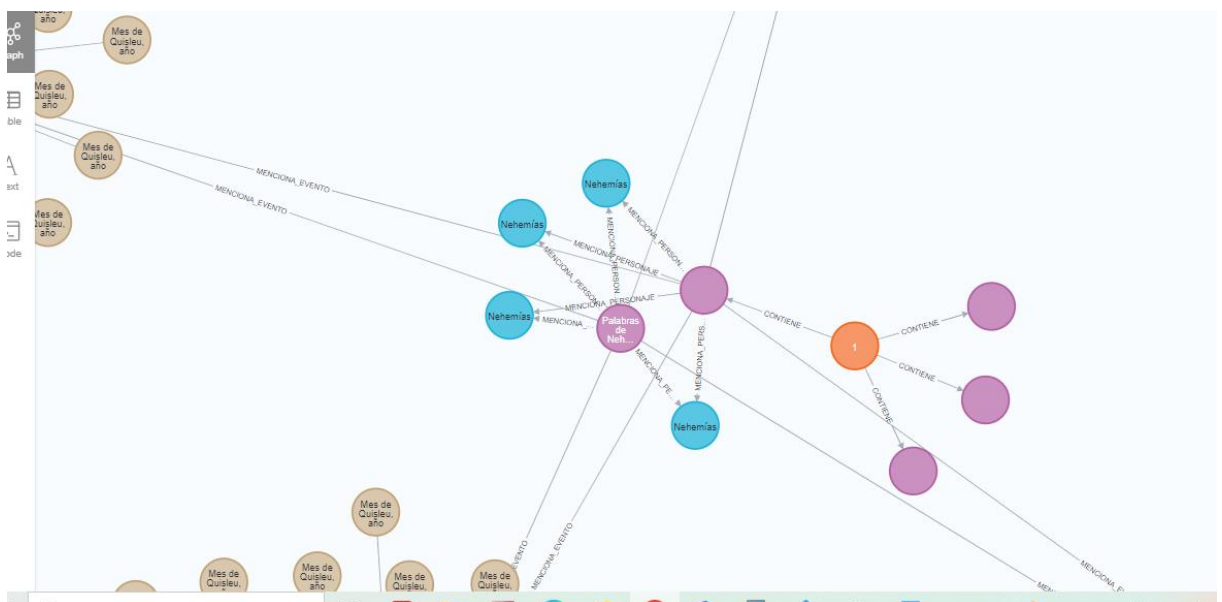


Figura 35 Otra vista ampliada de las relaciones del ejemplo la base de datos grafo Neo4j de libro de Nehemías

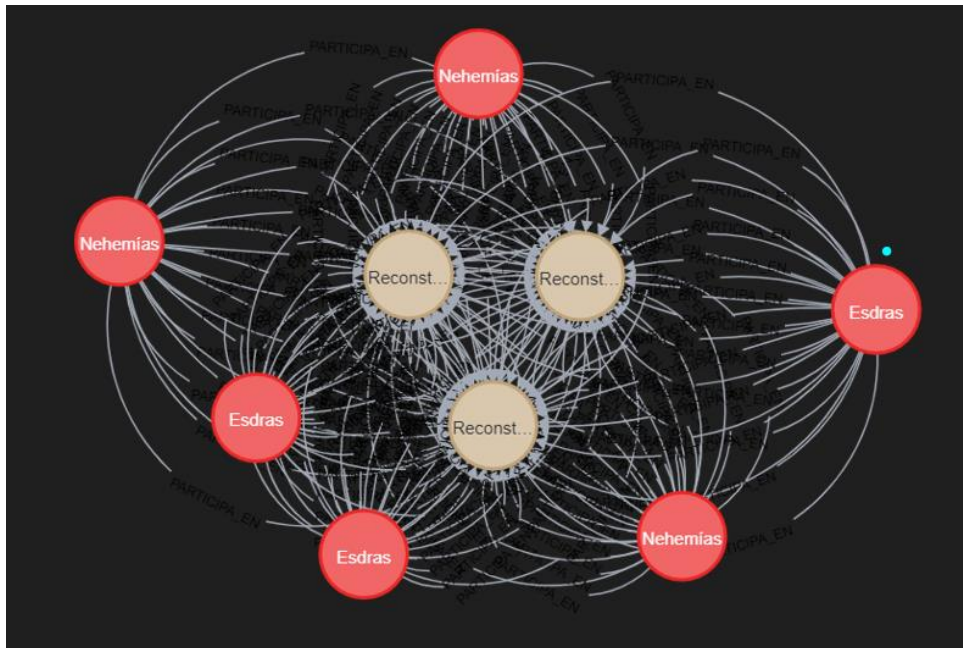


Figura 36 Vista de un ejemplo ampliado de los nodos y sus relaciones en el Libro de Nehemías con el personaje Esdras

Libros de la Biblia para el Piloto

Dentro de la clasificación de la biblia los tres últimos libros, Esdras, Nehemías y Ester, narran tiempos difíciles por los cuales pasa el pueblo de Israel. Estos libros narran el retorno del exilio a Palestina. Cada libro tiene un tópico específico, Esdras narra la reconstrucción del templo (460 y 440 a.C.), Nehemías (445-420 a.C.) reconstrucción de los muros del templo y Ester se desarrolla en la ciudad de Susa, capital del imperio persa y durante el reinado del rey Asuero o Jerjes (486-485 a.C.) registra la fiesta de Purim. (Suarez, 2013)

4.1. Descripción piloto experimental

Cuando se habla de “big data” se refiere a conjuntos de datos grandes y complejos que para procesarlos se necesitan herramientas que sean útiles para manejarlos eficientemente. La Biblia como un texto de datos a analizar, por ser una colección de textos religiosos, puede considerarse como un conjunto de datos, su clasificación como big data dependerá de la escala y el contexto de comparación, el volumen de datos que contienen un texto considerable. Cuando se compara 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 pero la complejidad del texto donde su lingüística, su valor espiritual, cultural e histórico la colocan en un buen ejemplo, en el ambiente del procesamiento de lenguaje natural (NLP) lo que conlleva a plantear desafíos y requerir técnicas avanzadas de análisis de texto

y aún más cuando se realiza un análisis detallado se necesita de enfoques y *herramientas* de procesamiento de grandes volúmenes de texto.

4.1.3 Participación y técnicas de evaluación.

El grupo de personas al cual está 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.

Como grupo de apoyo para revisar la usabilidad y desarrollo de la experiencia se trabaja con La Federación Bautista independiente de Colombia y sus miembros que realizaran las evaluaciones del prototipo (Anexo III). Los pastores que tienen años de experiencia y con estudios teológicos y están localizados en la ciudad de Bogotá y otras ciudades de Colombia.

Para la evaluación de los resultados se diseñan encuestas (Anexo I).

También para las revisiones de que biblias realizó otra encuesta (Anexo I Encuesta 2) de personas que son lideres y también con un grupo más general.

4.1.4. Cómo transcurrió el experimento.

Para la realización del prototipo piloto se utilizan herramientas que sean apropiadas que permitan trabajar con recursos colaborativos y que sean fuentes abiertas y totalmente gratis. Lo primero que se realizo fue crear un sitio en GitHub (Gonzalez, 2023) con el propósito de tener la documentación y el repositorio utilizando todas las características de este sitio.

Luego se creó un acceso a Google Colab sabiendo que para el piloto se está trabajando con código de lenguaje Python y sus librerías y que se tiene la ventaja de estar dirigida para la ciencia de datos utilizando recursos importantes de hardware gratuito de GPU.

La tercera parte unir o clonar el repositorio de GitHub en Google Colab (Anexo 2) La unión de estas dos plataformas abiertas y que se complementan es muy ventajoso para el proyecto ya que se puede tener conectado el desarrollo la dataset en ambientes totalmente abiertos.

El siguiente punto ha sido revisar las dataset que son públicas en lo que se encuentran que las bases de datos tienen diferentes formatos de almacenamiento con estructuras dadas por cada autor y que para utilizarlas se hace necesario hacer transformaciones a cada una en particular.

Con el punto anterior se hace necesario crear nuevas estructuras ya sea para adicionarlo a las bibliotecas de Python y generar una propia dirigida a generar una estructura para los libros

bíblicos de manera que se pueda dar la posibilidad de obtener información particular de la cronología de los eventos y los personajes de la Biblia. 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,

Se configuro un entorno para trabajar con la biblioteca spaCy de Python y así usar código para realizar operaciones de NLP para extraer información importante. Estas operaciones incluían Tokenización, lematización e identificación de relaciones sintácticas entre tokens individuales en una oración. Se necesita implementar un algoritmo para derivar los tokens necesarios de un árbol de dependencia, usando las características lingüísticas asignadas a los tokens.

4.2 Descripción de los resultados

Con la evaluación de los datasets como una forma de revisar el estado del arte en este tema se encuentran varios sitios que empezaron hace varios años y no están disponibles, otros ya no están en las páginas que se revisaron, la siguiente figura es una página (URL) que al ir a su sitio se encuentra un error 404 que significa que la pagina que se solicita ya no existe en el servidor. (Bible-json, n.d.)

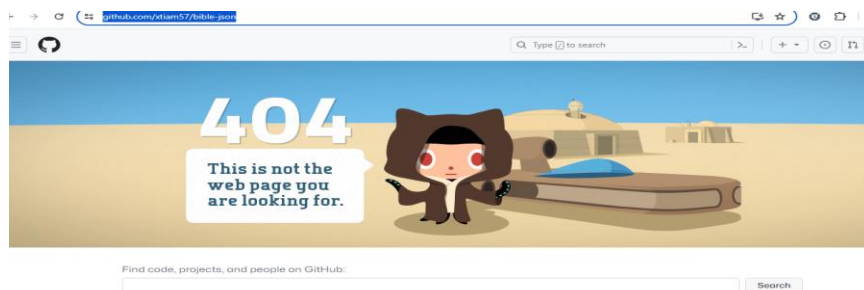


Figura 37 Error 404. EL URI esta incorrecto o no existe

Otros dataset tienen estructuras diferentes que hay que transformar o unir datos de texto en una solo dataset. La siguiente figura muestra un ejemplo (Romero, n.d.)

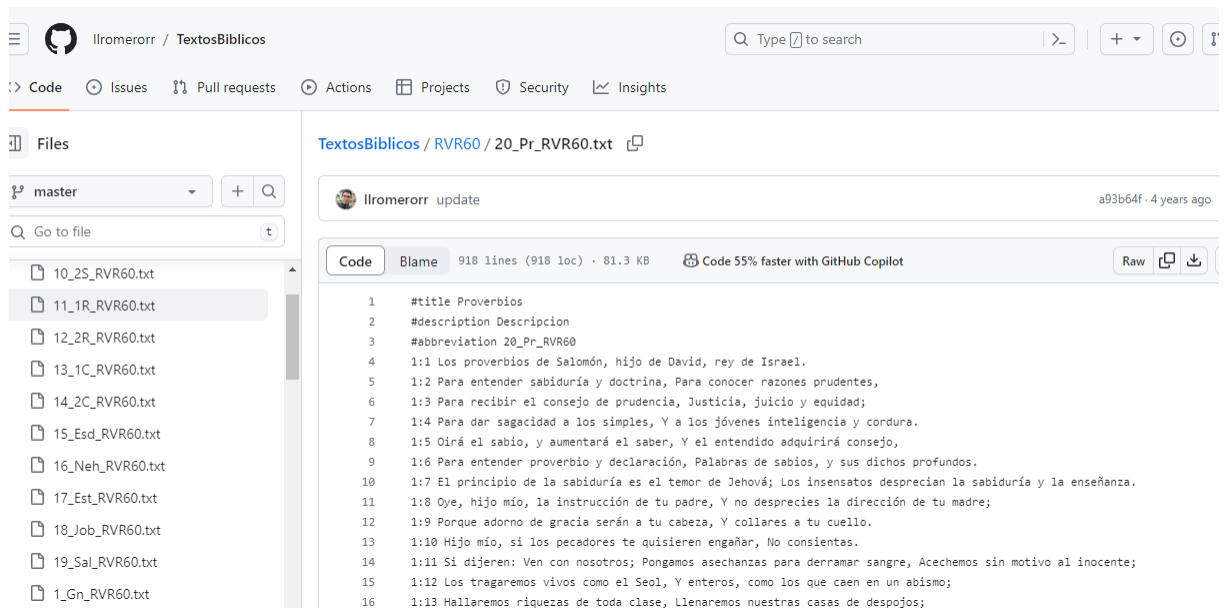


Figura 38 Dataset en GitHub con versiones de la biblia en texto

Con Google Colab se puede leer estos archivos utilizando la biblioteca request de Python, el código se deja documentado en GitHub, así se van haciendo las pruebas en el desarrollo de piloto experimental. La figura siguiente muestra el código ya grabado en GitHub. (Gonzalez, GitHub, n.d.)

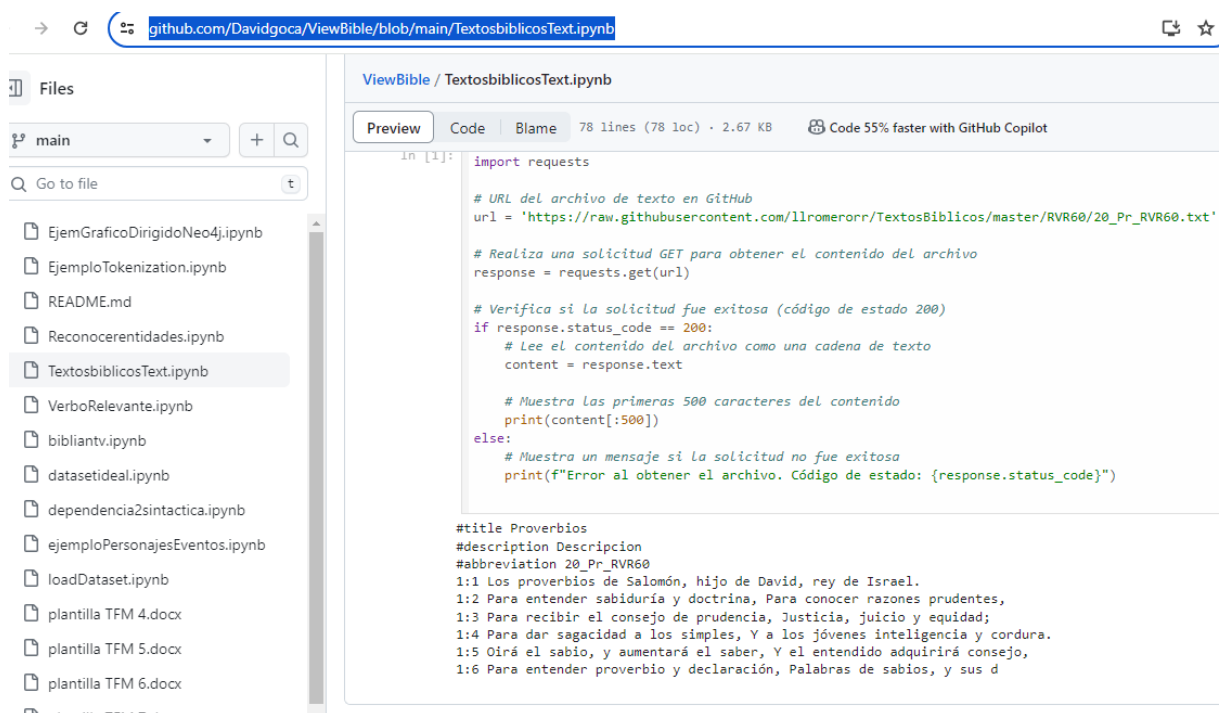


Figura 39 Pantallazo del sitio GitHub que contiene el código de Google Colab para leer una dataset texto

4.3. Discusión

Debido a la no consecución de una dataset completa, el primer punto que se contribuye es en este trabajo es presentar los requerimientos de la dataset utilizando Neo4j, por lo cual se definieron los nodos, las relaciones y las propiedades y con posibilidad de que vaya creciendo y mejorándose. Los ejemplos desarrollados las plataformas de Google Colab y GitHub son muy interesantes por la oportunidad de compartir el código y hacer una documentación sobre la investigación. El desarrollo del prototipo se logró avanzar en el modelo con Neo4j, Sin embargo, se reconoce la complejidad inherente de cada herramienta y se valora el dejar la base para continuar refinándolas en futuras investigaciones.

También el usar los libros de Esdras, Nehemías y Ester para empezar el prototipo, ha sido interesante porque las relaciones de los personajes, aparecen en diferentes tiempos, en otros libros y otras fechas aportando temas interesantes para los ejemplos propuestos.

5. Conclusiones y trabajo futuro

5.1. Conclusiones

Para el objetivo general de este piloto experimental de 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 las versiones públicas de la biblia se encontraron diferentes plataformas y tecnologías que conlleva a hacer transformaciones o integrarlas en uno solo formato. Adicionalmente, se necesitan completar los datos y así tener la posibilidad de hacer análisis a través de los tiempos.

Para realizar relaciones entre los personajes y eventos se desarrolla un modelo de una base de datos de grafos Neo4j. Con el lenguaje de consulta Cypher se realizan consultas descriptivas muy útiles que se pueden exportar en formatos JSON en siendo una buena plataforma para realizar análisis de texto que se llevan a herramientas colaborativas de GitHub y Google Colab. Allí Python con la biblioteca spaCy se puede hacer los diferentes pasos de análisis de texto.

Con la base de datos orientada a grafos Neo4j surge un punto crítico y es que para utilizarla en producción al no ser open source genera un compromiso económico que se debe discutir y aprobar por las organizaciones que asuman el liderazgo lo cual va a generar un beneficio de la comunidad cristiana.

La Contribución al campo académico es dejar sentada las bases para generar una base de datos no relacional que se pueda usar por los estudiosos de la biblia y utilizando NLP poder realizar análisis cronológicos y de relaciones entre personajes bíblicos y eventos a través del tiempo.

La investigación amplía el entendimiento existente sobre la cronología y las relaciones entre personajes bíblicos, y puede servir como punto de partida para investigaciones posteriores con implicaciones prácticas para estudiosos, líderes religiosos y comunidades de fe. Y pueden informar la enseñanza, la predicación o la interpretación de la Biblia en contextos académicos y religiosos.

Para desarrollar un prototipo de modelo NLP que permita extraer información cronológica de los textos bíblicos.

En el objetivo de crear una base estructurada se logra en la base de datos de grafos donde se trabajan los diferentes calendarios que aparecen en los textos bíblicos.

En cuanto al objetivo de utilizar herramientas de visualización para presentar los eventos y personajes en forma dinámica queda seguir desarrollando e integrando las herramientas como Power Bi integrándola con Neo4j.

5.2. Líneas de trabajo futuro

El tema tratado en la investigación es de una trascendencia en el ámbito cristiano que estudia y profundiza en temas que abre campos para utilizar las técnicas de inteligencia artificial en el campo de NLP combinándolas con las técnicas de estudio de textos antiguos que involucra temas como la hermenéutica y la exegesis.

Las líneas del trabajo a futuro son de continuar construyendo la base de datos que necesita incluir nodos que modelen la cronología bíblica generando patrones y conexiones que den una secuencia de los eventos que deben guardar una concordancia con el contexto histórico, cultural y religioso, unido a que cada evento tiene relaciones con personajes bíblicos.

Es muy conveniente que el proyecto se instale en plataformas colaborativas como GitHub y Google Colab, cuente con fuentes de financiación para cada fase en cuanto a la obtención de herramientas como Neo4j. Adicionalmente el conformar un grupo de trabajo interdisciplinario que trate temas de las herramientas tecnológicas de análisis de datos y temas de análisis de contenido.

Hay diferentes áreas que pueden desarrollar investigaciones futuras en temas de enfoques metodológicos para profundizar en los análisis en la cronología bíblica. Basados en el modelo ideal, el paso a seguir es completar las bases de datos de toda la biblia, antiguo y nuevo testamento.

Utilizar técnicas de análisis de redes sociales para visualizar y analizar las relaciones entre personajes bíblicos a lo largo del tiempo. Esto podría revelar patrones de interacción, influencia y conexión entre los diferentes individuos mencionados en la Biblia.

Realizar estudios comparativos entre la cronología bíblica y otras fuentes históricas contemporáneas para identificar coincidencias y discrepancias. Esto podría ayudar a contextualizar aún más los eventos bíblicos dentro del contexto histórico más amplio.

Se puede desarrollar aplicaciones pastorales y educativas que brinden recursos y materiales educativos basados en los hallazgos de tu investigación para ser utilizados en contextos pastorales, educativos y de enseñanza religiosa. Esto podría incluir guías de estudio, currículos o herramientas de enseñanza diseñadas para ayudar a las personas a comprender mejor la narrativa bíblica y su contexto histórico.

Otra línea de trabajo hacia el futuro puede ser entrenar datasets para investigar temas recurrentes en la narrativa bíblica y cómo estos temas se relacionan con la cronología y las relaciones entre personajes. Por ejemplo, podría explorarse cómo temas como la fe, la justicia o la redención se desarrollan a lo largo del tiempo en la Biblia. Profundizar en el análisis teológico de las relaciones entre personajes bíblicos y eventos, considerando cómo estas relaciones contribuyen a temas teológicos más amplios, como el plan de salvación, la providencia divina o la relación entre Dios y la humanidad.

Colaborar con expertos en disciplinas relacionadas, como la arqueología, la historia antigua, la lingüística y la teología, para abordar preguntas interdisciplinarias sobre la cronología y las relaciones entre personajes bíblicos. Esta colaboración puede enriquecer el análisis y proporcionar nuevas perspectivas sobre el tema.

Otra línea de trabajo futuro es poder crear con Power BI visualizaciones interactivas basadas en los datos almacenados en Neo4j, diseñar paneles y tableros que permita explorar y comprender mejor las relaciones de los eventos y personajes a través del tiempo.

Bibliografía

- A webcomic of romance*,. (n.d.). Retrieved from vuelve a golpear el clavo con el martillo
- Alvear, J. (1995). Cristianismo Primitivo y Religiones Místicas. In J. Alvear, *Cristianismo Primitivo y Religiones Místicas* (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/>
- Bernardino, D. F. (2018). Graph Databases Comparison: AllegroGraph, ArangoDB, InfiniteGraph, Neo4J, and OrientDB . Retrieved from <https://www.scitepress.org/papers/2018/69102/pdf/index.html>
- Bible-json*. (n.d.). Retrieved from <https://github.com/xtiam57/bible-json>
- 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
- Broecheler, D. K. (2020). *The Practitioner's Guide Graph Data*. O'Reilly Media Inc.
- 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/>
- Christodoulopoulos. (n.d.). *A multilingual parallel corpus created from translations of the Bible*. Retrieved 202, from <https://github.com/christos-c/bible-corpus>
- Coghlan, N. (2014, 01 06). *Python notes*. Retrieved 2024, from https://python-notes.curiousericiency.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.
- DB-ENGINES RANKINGS. (n.d.). Retrieved 01 28, 2024, from <https://db-engines.com/en/ranking>
- 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/>
- Gonzalez, D. (2023, 12 01). <https://github.com/Davidgoca/personajesyeventosbiblicos>. (D. Gonzalez, Editor) Retrieved from <https://github.com/Davidgoca>
- Gonzalez, D. (n.d.). *GitHub*. Retrieved from <https://github.com/Davidgoca/ViewBible/blob/main/datasetideal.ipynb>
- Gonzalez, D. (n.d.). *GitHub*. Retrieved from <https://github.com/Davidgoca/ViewBible/blob/main/TextosbiblicosText.ipynb>
- 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*.
- Luhn, 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/>
- Romero, L. (n.d.). *GitHub*. Retrieved from https://github.com/llromerorr/TextosBiblicos/blob/master/RVR60/20_Pr_RVR60.txt
- 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 lirture* . 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.

- spaCy. (n.d.). Retrieved from spaCy: <https://spacy.io/api>
- Srinivasa-Desikan, B. (2018). In B. Srinivasa-Desikan, *Natural Language Processing and Computacional Linguistics* (pp. 9-23).
- Suarez, B. C. (2013). *Libro a Libro por el antiguo Testamento*.
- 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>
- Transformers Hugging 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 Encuestas

Encuesta 1

Se realizó la encuesta utilizando Google Forms

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

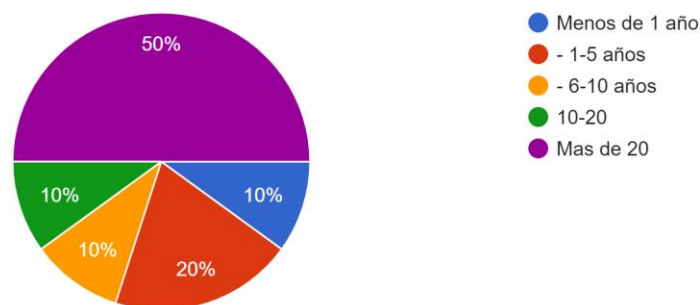
Estimado Pastor

Es un placer para nosotros invitarlo a 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.

¿Cuántos años de experiencia tienes como pastor cristiano?

10 responses



¿En qué denominación sirves como pastor?

10 responses

Bautista
Federación bautista de Colombia
Bautista del sur
Bautista independiente
Bautista

Iglesia Bautista Fundamental el Calvario
Iglesia Bautista Valle de Josafat
Federación Bautista independiente de Colombia
Bautista de la Serena

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

11 responses



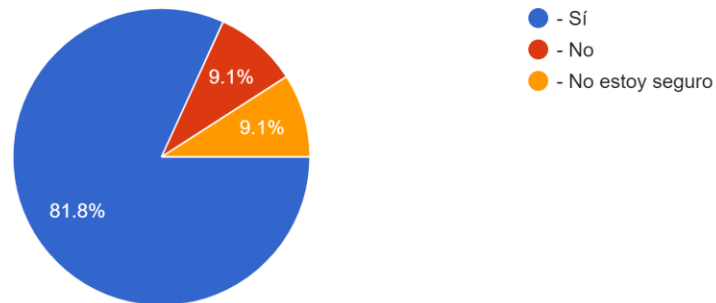
¿Cuáles son tus libros bíblicos favoritos para enseñar?

11 responses

1 Juan, Romanos y Filipenses
Toda la biblia estudio sistemático
Mateo
1 Juan y los Evangelios
No tengo favorito para enseñanza
NEHEMIAS
Génesis, Históricos, Eclesiastés, Daniel, Profetas Menores, Lucas, Hechos, 2 Timoteo, Judas
Teología
Hechos
Nehemías, Rut, epístolas de Juan
Nehemías

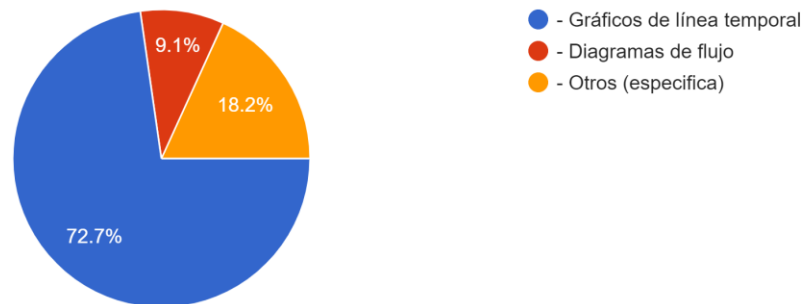
¿Utilizas herramientas de visualización para ayudarte a comprender mejor la Biblia y preparar tus enseñanzas?

11 responses



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

11 responses



¿Cuál es tu personaje bíblico favorito?

11 responses

Daniel
Pablo
David
JOSÉ
DANIEL
José, el hijo de Jacob
Daniel y José
Pablo
Nehemías

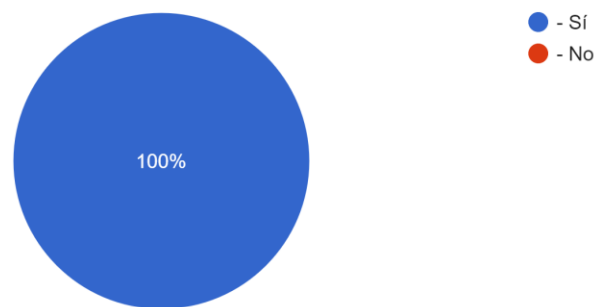
¿Hay algún evento bíblico en particular que encuentres especialmente significativo para tu ministerio?

10 responses

Muerte de Moisés y nombramiento de Josué cómo sucesor
La creación
Participación se seminarios
El Éxodo
SALVACION
La dependencia de José a Dios en medio de todo tipo de circunstancias: "pero Jehová estaba con José..."
Vida de José
Hechos 19;1
Si cuando Nehemías es perseguido por sus enemigos. Sambalat Tobías.
Salvación

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

11 responses



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

11 responses

Ayudaría a afianzar el conocimiento y expresarlo de una manera más clara.
La enseñanza con audiovisuales me ayuda no solamente a mí sí no a la iglesia una imagen vale más que mil palabras
Ni se
La ilustración siempre será una herramienta que ayuda a la objetividad
Brindándome mayor claridad espacio temporal de los personajes y eventos, eso facilitaría encontrar relaciones.
334
Porque ayuda a ubicarse en el tiempo y que sucedieron las cosas, así como un mapa ayuda en ubicarse geográficamente
Mejor exposición

Sería para una explicación más exacta
Mostrando en tiempo real y la cultura se entenderá mejor la Palabra de Dios.
En el contexto

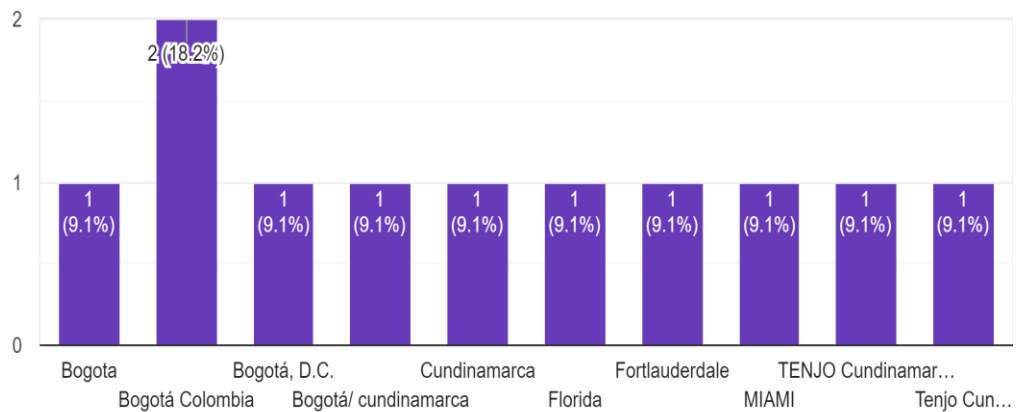
¿Tienes alguna sugerencia o comentario adicional sobre cómo mejorar el estudio de la Biblia mediante herramientas de visualización?

10 responses

No.
Gracias a Dios hoy día hay buen material sería muy bueno tener La oportunidad de obtener material
Ninguna
Clarificando la diferencia de personajes con el mismo nombre.
GJ
Mapas en 3d de los lugares Bíblicos con animación su fuere posible.
Archivos visuales por temas
No
Con cuadros sinópticos

Municipio / Departamento

11 responses



Encuesta 2

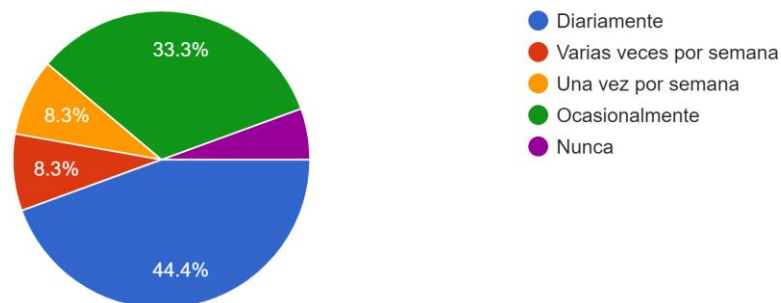
Se realizó la encuesta utilizando Google Forms

https://docs.google.com/forms/d/e/1FAIpQLSebH6IRJ0keiSrI0xeY86MO3If0c5erC2BAD17Tar-YZX1A/viewform?usp=sf_link

Las siguientes son las respuestas:

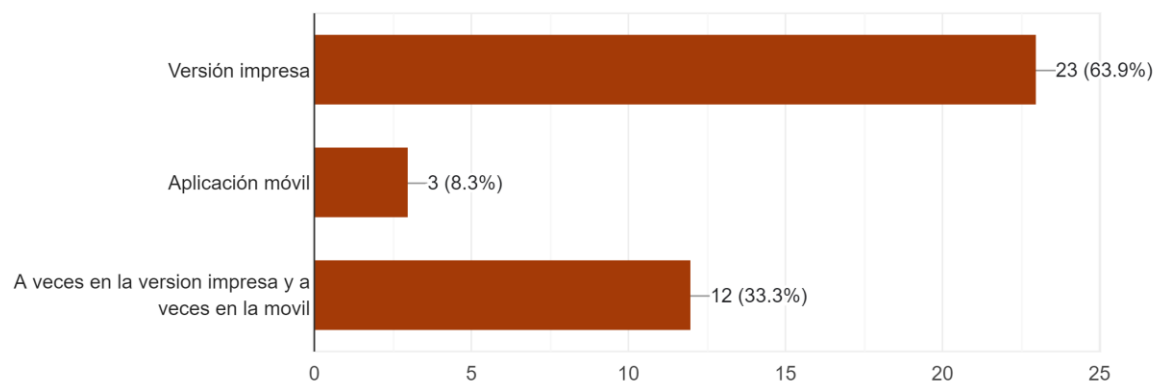
¿Con qué frecuencia lees la Biblia?

36 responses



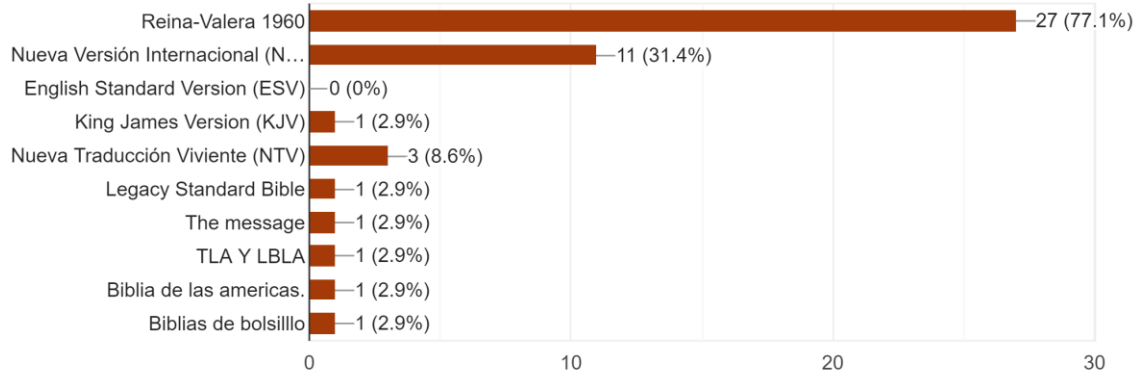
¿Prefieres leer la Biblia a través de la versión impresa o mediante una aplicación móvil?

36 responses



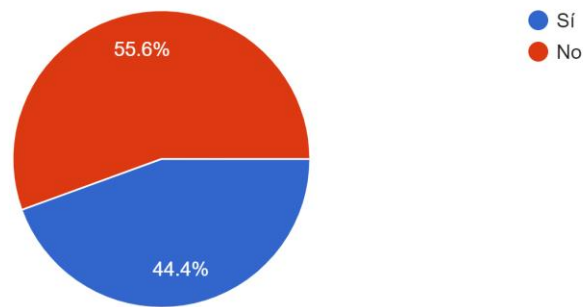
¿Qué versión/es de la Biblia utilizas con mayor frecuencia ? (Puedes seleccionar más de una opción)

35 responses



¿Utilizas alguna biblia de estudio?

36 responses

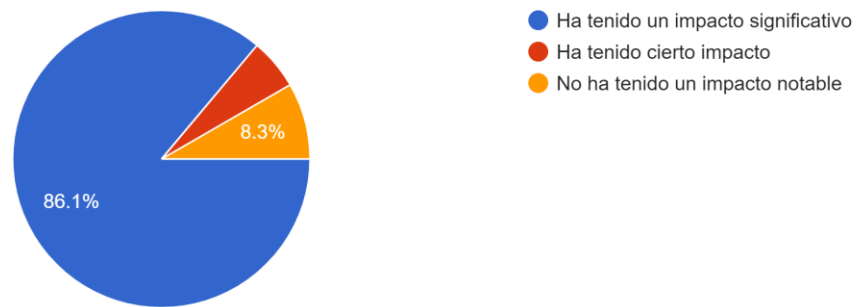


Al ser positiva la respuesta anterior. ¿Cuál versión de biblia utilizas o si tienes una biblia de estudio?

Razones espirituales
Interés académico
Thomson
MacArthur
Thomson
Reina Valera.
Reina Valera
Thompson, Espurgeon
V
Biblio inspira
THOMPSON
Valera
La Biblia Jesús
Diario vivir y la Thompson.
Obsequio por EMAÚS. EMAÚS
Reina Valera
MacArthur
No aplica

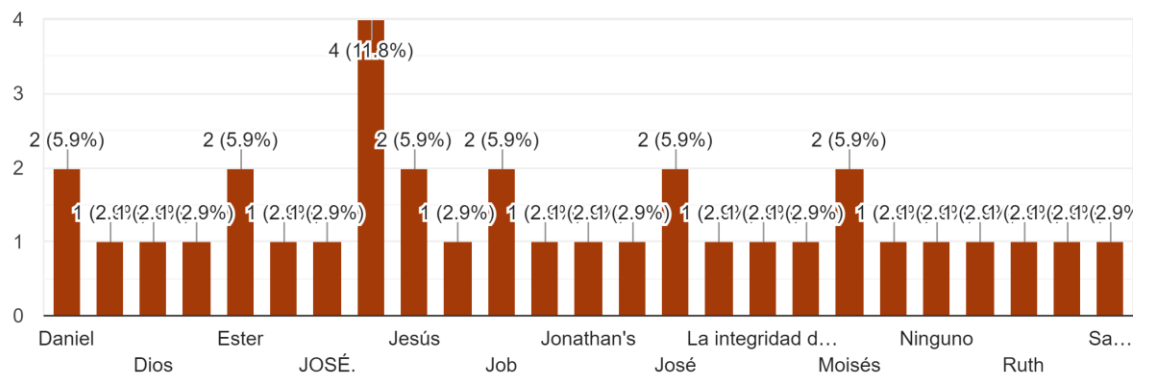
¿En qué medida consideras que leer o estudiar de la Biblia ha impactado tu vida?

36 responses



Tienes un personaje Bíblico favorito? Cual?

34 responses



¿Tienes un evento Bíblico favorito en el Antiguo Testamento? ¿Cual?

Reconstrucción de los muros de Jerusalén
El pueblo de Israel cruzando el Mar Rojo
José faraón
Salida de Egipto de Israel
La oposición a Nehemías por Sambalat y sus amigos
Shadrach, Meshach, Abed-nego se niegan a adorar a un ídolo
DAVID.... Un hombre Conforme al Corazón de DIOS.
La salida del pueblo de Israel y el cuidado que Dios tuvo con su pueblo.
Si, cuando Dios abre el mar en dos
Las plagas de Egipto
La barca y la fe
Dios
Daniel
Toda la conversación. de Job con sus amigos y Dios. Creo que se puede evidenciar la fe tan impresionante que tenía este personaje, pero también visualizar cómo en su humanidad ve a Dios, y el procesos y progreso que tiene con Él.

El diluvio
salmos
David en la cueva de Adulam
La historia de Rut
Genesis
El ascenso al monte SINAI
El arca de Noé
Los profetas de Baal en el monte Carmelo
La victoria que Dios dio a Elías con los baales
No tengo uno favorito. Me gusta la historia de Jonás, la de Job, la de Ruth, los milagros que hizo Eliseo,
no
Ninguno
La vida de José, el desprecio de sus hermanos
Historia de José
El diluvio universal
Jehová es mi pastor
Apocalipsis

This content is neither created nor endorsed by Google. [Report Abuse](#) - [Terms of Service](#) - [Privacy Policy](#)

_Forms

Anexo II Código de Python

Código 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 III Federación Bautista independiente de Colombia.

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

Anexo IV Datasets

A. Datasets que no tienen contenido

Conjunto de recursos cristianos para programadores.

Biblia Reina Valera 1960

Biblia organizada cada capítulo en un archivo de `JSON` que contiene un arreglo de arreglos, cada arreglo representa un capítulo. Gracias a <https://github.com/xtiam57/bible-json>.

Himnos Bautistas

(Bible-json, n.d.)

B. Datasets bíblicos que se revisaron en las diferentes plataformas abiertas.

Dataset en la plataforma Kaggle:

The screenshot shows the Kaggle search results for the query 'biblia'. The URL in the browser is <https://www.kaggle.com/search?q=biblia>. The search results are filtered by 'Relevance' and show 8 results. The first result is 'Biblia' by Paulo Gladson, which is a dataset of 6 Versões da Bíblia em Português Brasil, with 37 downloads. The second result is 'Biblia NTV (Spanish Bible NTV)' by Camilo Rold, which is a dataset of 6 Versões da Bíblia em Português Brasil, with 151 downloads. The third result is 'Bible Corpus' by Ozwin Rahediyen Hartono, which is a dataset of 6 Versões da Bíblia em Português Brasil, with 4,716 downloads. The fourth result is 'Biblia Entidades Decompostas' by Paulo Gladson, which is a dataset of 6 Versões da Bíblia em Português Brasil, with 0 downloads. The fifth result is 'Cross Lingual Information Retrieval using BERT' by Tarun Jagdish, which is a notebook, with 0 comments. The sixth result is 'Cria base da Biblia com Análise de Sentimentos' by Paulo Gladson, which is a notebook, with 0 comments. The seventh result is 'Estudo Biblico' by Paulo Gladson, which is a notebook, with 2 comments. The left sidebar shows filters for Creator, Dataset Size, Dataset File Types, Dataset License, Notebook Language, and Tags.

| Sitio | Resultado |
|--|------------------------------|
| 6 Versões da Bíblia em Português Brasil https://www.kaggle.com/datasets/paulogladson/biblia | La dataset está en portugués |

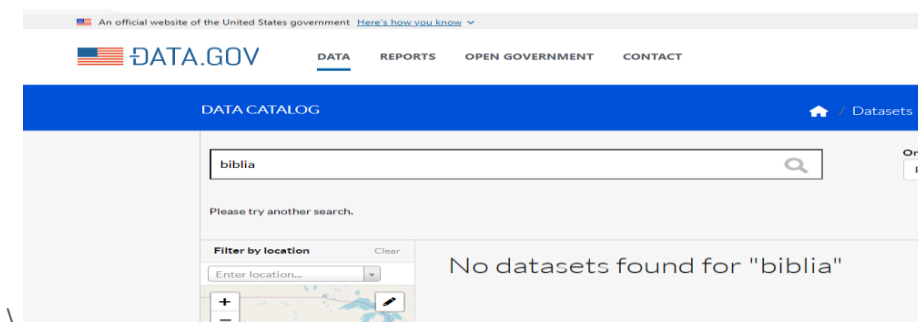
| Sitio | Resultado |
|---|--|
| Bible Corpus: Traducción de la biblia en ingles para versiones de SQL, SQLite, XML formato y Jason https://www.kaggle.com/datasets/oswinrh/bible | Esta solo en inglés. |
| Traducción Viviente tomada https://www.kaggle.com/datasets/comesruiz/biblia-ntv-spanish-bible-ntv/data | En español y su estructura en csv que contiene el nombre del libro, capítulo, versículo y texto. |
| Biblia Entidades Decompostas https://www.kaggle.com/datasets/paulogladsen/biblia-entidades-decompostas | En idioma de de compostas |
| Estudio Bíblico Python · <u>Biblia</u> https://www.kaggle.com/code/paulogladsen/estudo-4AQ | Errores de ejecución |
| Biblia com Análise de Sentimento https://www.kaggle.com/datasets/paulogladsen/biblia-sentiment | |
| Explore King James Bible Books https://www.kaggle.com/code/gpreda/explore-king-james-bible-books | En ingles |
| The King James Version of the Bible https://www.gutenberg.org/ebooks/10 | Biblia de Gutenberg en ingles |
| Reina Valera New Testament of the Bible 1858 by Reina and Valera https://www.gutenberg.org/ebooks/5878 | |
| Reina Valera New Testament of the Bible 1909 by Reina and Valera https://www.gutenberg.org/ebooks/5881 | |
| BibleData The Bible in Structured Data ...download estructura enc vs personajes y eventos | |

| Sitio | Resultado |
|---|---|
| https://www.kaggle.com/datasets/bradystephenson/bibledata?select=BibleData-Event.csv | |
| Biblia NTV (Spanish Bible NTV) Nueva Traducción viviente https://www.kaggle.com/datasets/comesruiz/biblia-ntv-spanish-bible-ntv | Biblia en español versión Nueva Traducción Viviente tomada Archivo CSV que contiene el nombre del libro, capítulo, versículo y texto. |
| https://www.kaggle.com/code/jjmewtw/total-bible-text-study-eda-cluster-bert-nlp | Esta en Ingles |

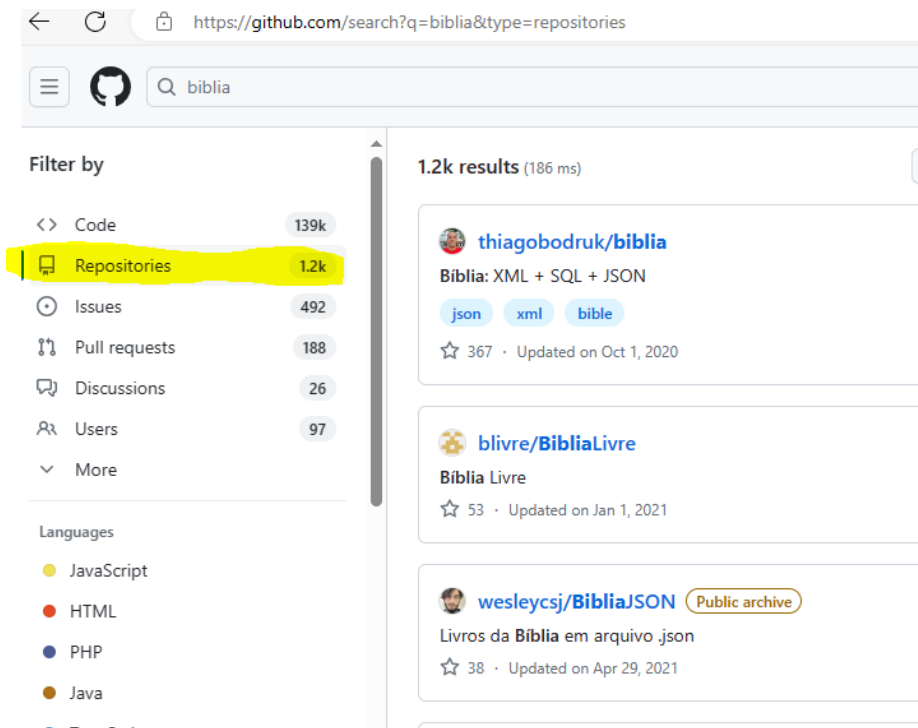
Datasets en BiblIA

| | |
|---|--|
| BiblIA - an Open Annotated Dataset BiblIA - an Open Annotated Dataset (zenodo.org) | Este conjunto de datos para el reconocimiento de texto escrito a mano incluye segmentación de diseño (regiones, líneas superiores y polígonos de línea) y transcripciones Unicode imágenes de manuscritos hebreos medievales |
|---|--|

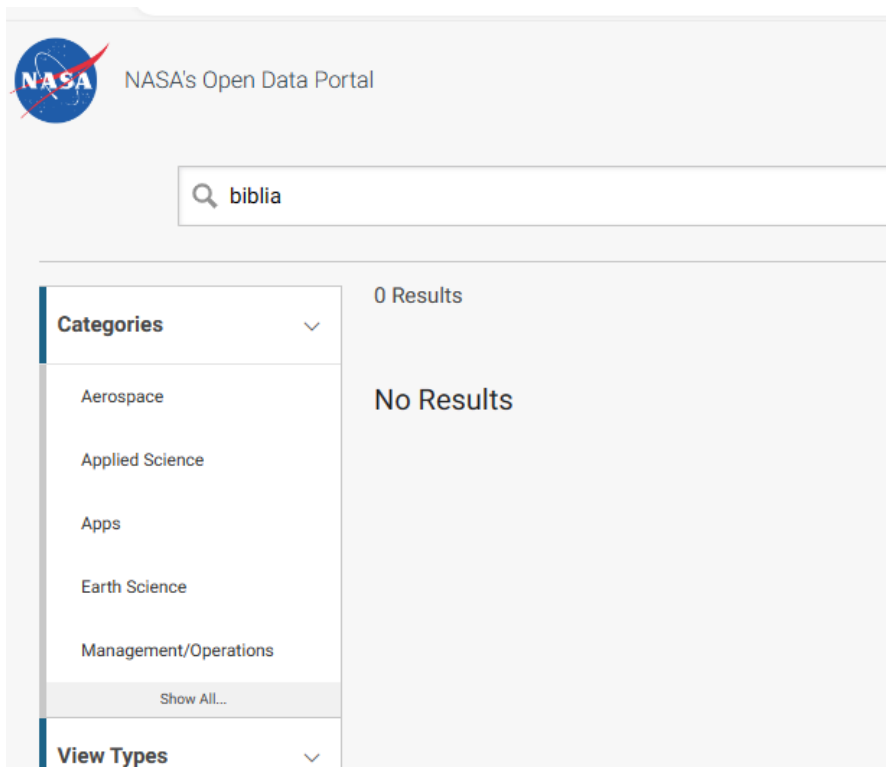
Dataset en la página de data del gobierno de USA.



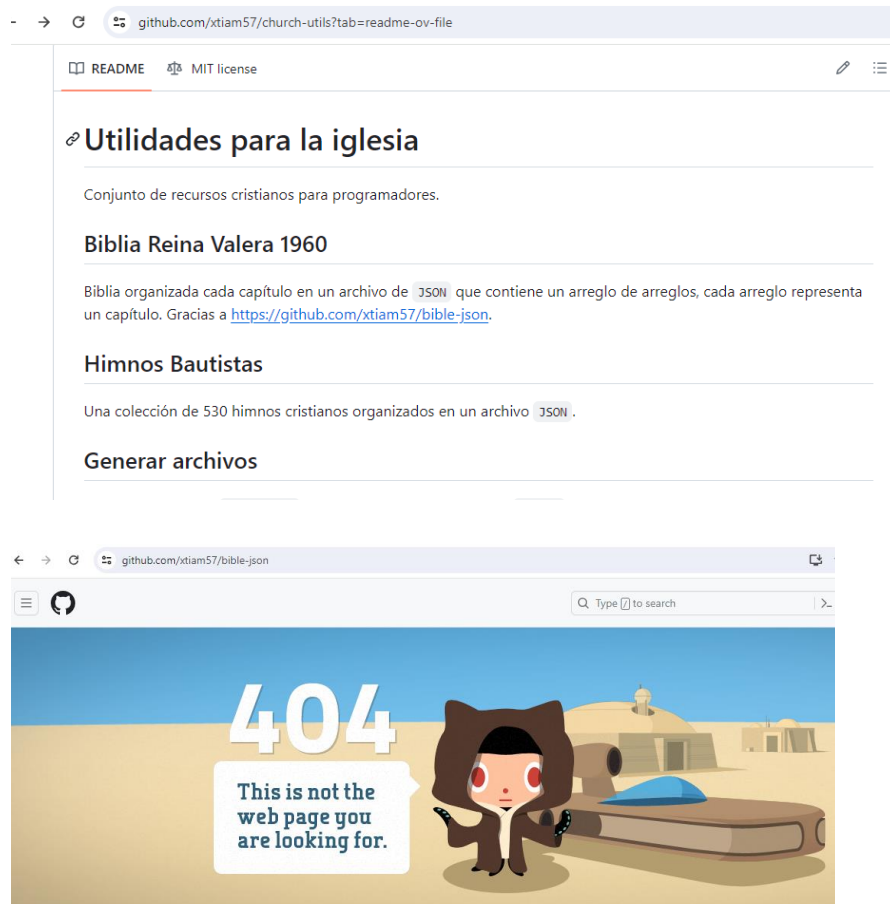
Datasets en GitHub



Dataset en el webpages de la Nasa

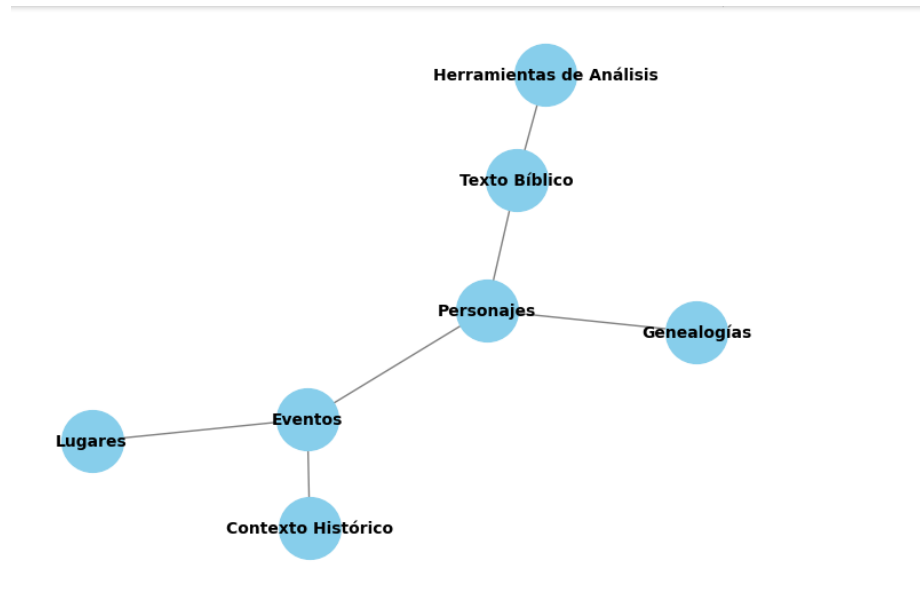


En los datasets se encontraron sitios con base de datos relacionales y también en muchos casos se encontró que las páginas ya no están o son proyectos que empezaron y están sin continuar o en desarrollo como se muestra en los pantallazos siguientes



Anexo V Dataset Ideal

Grafica realizada con código Python



```
import networkx as nx
import matplotlib.pyplot as plt

# Crear un grafo
G = nx.Graph()

# Agregar nodos y conexiones
G.add_nodes_from(["Texto Bíblico", "Personajes", "Eventos", "Lugares", "Genealogías", "Contexto Histórico", "Herramientas de Análisis"])
G.add_edges_from([("Texto Bíblico", "Personajes"), ("Personajes", "Genealogías"), ("Personajes", "Eventos"),
                  ("Eventos", "Lugares"), ("Eventos", "Personajes"), ("Contexto Histórico", "Eventos"),
                  ("Herramientas de Análisis", "Texto Bíblico")])

# Visualizar el grafo
pos = nx.spring_layout(G) # Configuración del diseño del grafo
nx.draw(G, pos, with_labels=True, font_weight='bold', node_color='skyblue', node_size=1500, edge_color='gray', linewidths=1, font_size=10)

# Mostrar el grafo
plt.show()
```

Se utilizó la biblioteca NetworkX para crear un grafo sencillo y matplotlib para visualizarlo. El código queda revisado en el sitio GitHub donde se ha trabajado la investigación.

(Gonzalez, GitHub, n.d.)

Anexo VI Modelo de base de datos de grafo

Modelo de base de datos de grafo con Neo4j para la Biblia. Se define nodos y relaciones que representen los elementos clave, como libros, capítulos, versículos, personajes y eventos.

Nodos:

Libro: Cada libro de la Biblia puede ser representado como un nodo con la etiqueta `Libro`. Podrías asignar propiedades como `nombre` para el nombre del libro y `testamento` para indicar si pertenece al Antiguo o al Nuevo Testamento.

Capítulo: Cada capítulo de un libro puede ser un nodo con la etiqueta `Capitulo`. Podrías asignar una propiedad `numero` para indicar el número del capítulo.

Versículo: Cada versículo dentro de un capítulo puede ser un nodo con la etiqueta `Versiculo`. Podrías asignar una propiedad `numero` para indicar el número del versículo.

Personaje: Cada personaje bíblico puede ser representado como un nodo con la etiqueta `Personaje`. Puedes asignar propiedades como `nombre` y `rol` para describir al personaje.

Evento: Cada evento importante puede ser un nodo con la etiqueta `Evento`. Puedes asignar propiedades como `nombre` y `fecha` para describir el evento.

Relaciones:

Se puede establecer relaciones entre estos nodos para representar las conexiones. Por ejemplo, se puede tener relaciones que conecten un Libro con sus Capítulos y los Capítulos con sus `Versículos`. También podrías tener relaciones que conecten `Personajes` con `Eventos` en los que participaron.

En código Cypher seria:

```
(: Libro {nombre: "Génesis", testamento: "Antiguo Testamento"})  
  
(: Capitulo {numero: 1})  
  
(:Versiculo {numero: 1, texto: "En el principio creó Dios los cielos y la tierra."})  
  
(: Personaje {nombre: "Moisés", rol: "Profeta"})  
  
(: Evento {nombre: "Éxodo", fecha: "Siglo XIII a.C."})  
  
// Relación entre Libro y Capitulo
```

(: Libro)-[: TIENE]->(: Capitulo)

// Relación entre Capitulo y Versículo

(:Capitulo)-[:CONTIENE]->(:Versiculo)

// Relación entre Personaje y Evento

(:Personaje)-[:PARTICIPA_EN]->(:Evento)

Anexo VII Código Cypher

Ejemplo para crear el libro de Nehemías.

// Crear los versículos del capítulo 1 de Nehemías con su texto

CREATE (v1:Versiculo {numero: 1, texto: 'Palabras de Nehemías hijo de Hacalías. Aconteció en el mes de Quisleu, en el año veinte, estando yo en Susa, capital del reino,'})

CREATE (v2:Versiculo {numero: 2, texto: 'que vino Hanani, uno de mis hermanos, con algunos hombres de Judá. Y les pregunté por los judíos que habían escapado, que habían quedado de la cautividad, y por Jerusalén.'})

CREATE (v3:Versiculo {numero: 3, texto: 'Y me dijeron: El remanente, los que quedaron de la cautividad, allí en la provincia, están en gran mal y afrenta, y el muro de Jerusalén derribado, y sus puertas quemadas a fuego.'})

CREATE (v4:Versiculo {numero: 4, texto: 'Aconteció que cuando oí estas palabras, me senté y lloré, y estuve de duelo por algunos días, y ayuné y oré delante del Dios de los cielos,'})

// Crear el capítulo 1 de Nehemías y asociar los versículos

CREATE (capitulo1:Capitulo {numero: 1})

WITH capitulo1

UNWIND [1, 2, 3, 4] AS verseNumber

CREATE (v:Versiculo {numero: verseNumber})

CREATE (capitulo1)-[:CONTIENE]->(v)

// Crear los personajes y eventos relacionados

CREATE (:Personaje {nombre: 'Nehemías'})

CREATE (:Personaje {nombre: 'Hanani'})

CREATE (:Evento {nombre: 'Conversación con Hanani'})

CREATE (:Evento {nombre: 'Reconstrucción del muro de Jerusalén'})

// Asociar los personajes y eventos a los versículos pertinentes

WITH 1 AS ignored

MATCH (v1:Versiculo {numero: 1})

MATCH (nehemias:Personaje {nombre: 'Nehemías'})

MATCH (hanani:Personaje {nombre: 'Hanani'})

MATCH (conversacion:Evento {nombre: 'Conversación con Hanani'})

MERGE (v1)-[:MENCIONA_PERSONAJE]->(nehemias)

MERGE (v2)-[:MENCIONA_PERSONAJE]->(hanani)

MERGE (v1)-[:MENCIONA_EVENTO]->(conversacion)

// Asociar la fecha del evento 'Conversación con Hanani' al versículo 1

CREATE (:Fecha {nombre: 'Mes de Quisleu, año veinte'})-[:OCURRE_EN]->(conversacion)
