UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS
DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN

TEXTARIUM: UN SISTEMA DE VISUALIZACIÓN DE TEXTOS UTILIZANDO CONOCIMIENTO LIBRE Y PROCESAMIENTO DE LENGUAJE NATURAL

MEMORIA PARA OPTAR AL TÍTULO DE
INGENIERO CIVIL EN COMPUTACIÓN

SEBASTIÁN IGNACIO SALINAS RODRÍGUEZ

PROFESOR GUÍA:
EDUARDO GRAELLS GARRIDO

MIEMBROS DE LA COMISIÓN:
ANDRÉS ABELIUK KIMELMAN
JAZMINE MALDONADO FLORES

SANTIAGO DE CHILE
2023

Resumen

La literatura y los textos no literarios han desempeñado un papel trascendental en la evolución de las sociedades humanas a lo largo de la historia. La influencia de ambos ha sido innegable en diversos ámbitos, tales como la cultura, la educación, la política, el entretenimiento, entre otros. Gran parte de este éxito se debe a la utilización de ilustraciones para representar y transmitir de manera más efectiva los mensajes que se desean comunicar.

Sin embargo, los procesos creativos y de redacción asociados a la escritura de un texto que transmita fielmente la información, ideas, conocimientos, experiencias o sensaciones del autor, conllevan una serie de desafíos y complicaciones que requieren una inmensa cantidad de esfuerzo y dedicación para ser superados con éxito.

Es en este contexto que surge la idea y la oportunidad de apoyar a los escritores en la superación de obstáculos y problemáticas comunes de los procesos de redacción, como los bloqueos creativos, la falta de inspiración, las dificultades para expresarse con precisión, entre otras. Esto por medio de una herramienta que brinde apoyo y recursos que faciliten la generación de contenido, proporcionando una ventaja significativa en la creación de textos.

Con esto en mente, se plantea la factibilidad de desarrollo del sistema *Textarium*, una herramienta de análisis y visualización de texto que facilita la identificación de palabras clave, el reconocimiento de entidades y las relaciones entre los elementos del texto, además de ofrecer apoyo ilustrativo para enriquecer el relato que se esté analizando. Todo esto integrado en forma de una aplicación accesible e intuitiva que potencie la creatividad de los usuarios y que brinde un soporte eficaz al proceso de escritura.

El proyecto ha alcanzado su objetivo al demostrar la factibilidad de la herramienta a través del desarrollo de un producto mínimo viable (*MVP*) centrado en el aprovechamiento de técnicas de Procesamiento del Lenguaje Natural (*NLP*), modelos grandes del lenguaje (*LLMs*) como *BERT* y modelos generativos como *Stable Diffusion*, logrando generar contenido adicional y que aporte valor a las descripciones de textos analizadas por el sistema.

Una ventaja distintiva del proyecto *Textarium* es su enfoque en componentes modularizadas y uso de software libre, lo que promueve su accesibilidad y colaboración. La aplicación desarrollada puede ser adoptada y modificada por una comunidad amplia de desarrolladores y usuarios, lo que potencia su utilidad y flexibilidad. Con esta iniciativa, se logra potenciar la industria creativa al ofrecer a diversos usuarios las herramientas necesarias para perfeccionar sus trabajos y alcanzar nuevos niveles de excelencia en sus creaciones.

A mi familia, amigos y a todos los que creyeron en mí.

Agradecimientos

Quiero partir agradeciendo con mucho cariño a mis padres, a mis hermanos, a mis abuelos, a mis tíos, y en general a todos los miembros de mi familia que me han acompañado durante este proceso y que han formado parte de mi vida desde que llegué a este mundo hasta el punto en que me encuentro ahora, nada de esto hubiera sido posible sin su apoyo y amor incondicional, gracias por tanto.

También agradezco mucho a la Marita, las vueltas de la vida nos juntaron en este camino llamado DCC, volviéndolo más ameno y enriqueciendo el *lore* y desarrollo de personaje que me esperaba en esta etapa. Gracias por todo, por tu amor, tu compañía, tu sentido del humor, tus mañas, por confiar en mí y por estar ahí siempre que lo necesité, te amo un montón.

Por supuesto que no puedo olvidarme de los 31G. Nunca logramos la meta de llegar a ser 31, pero mejor, ya que si los gramos fueran billetes, al menos no tendría 31 de ellos falsos.

Tampoco me olvido de los Civil Panas, del Vicho, de Rodolfo y tantos otros que han hecho de mi vida universitaria un recuerdo que llevaré siempre conmigo. A todos y todas muchas gracias por los carretes, viajes y momentos que vivimos juntos.

Gracias también a los cabros de *Inventures* por su interés genuino y por todas las facilidades que me dieron cuando me encontraba en mis peores momentos.

A mis amigos y amigas del colegio, a quienes he dejado un poco de lado en esta travesía, pero uno siempre vuelve a donde fue feliz.

A mi profesor guía, que me acompañó y ayudó con un montón de recomendaciones e ideas a lo largo de los intensos meses en los que trabajé en esta memoria.

Y finalmente, pero no menos importante, a todos los que se acercaron desinteresadamente a este trabajo y que contribuyeron de una u otra forma, muchas gracias a todos.

Tabla de Contenido

1. Introducción **1**

1.1. Contexto 1

1.2. Motivación 3

1.3. Objetivos 5

1.3.1. Objetivo General 5

1.3.2. Objetivos Específicos 5

1.4. Metodología 6

1.5. Estructura del Documento 7

2. Marco Teórico **8**

2.1. Antecedentes 8

2.1.1. Redes Neuronales Artificiales 9

2.1.2. Deep Learning 11

2.2. Procesamiento del Lenguaje Natural 12

2.2.1. Áreas y Fundamentos del *NLP* 13

2.2.2. Procesamiento de Texto con Redes Neuronales 15

2.2.3. *Transformers* 18

2.2.4. Modelos del Lenguaje 20

2.3. Modelos Generativos 22

2.3.1. Modelos de Difusión 23

2.4. Trabajos Relacionados 25

3. Diseño de la Solución **27**

3.1. Requisitos de Software 28

3.2. Arquitectura General 29

3.2.1. Módulo de Preprocesamiento 31

3.2.2. Módulo de Extracción de Conceptos Claves 32

3.2.3. Módulo de Detección de Entidades 32

3.2.4. Módulo de Ilustración 32

3.3. Integración e Interfaz 32

4. Implementación de la Solución **33**

4.1. Primeros Pasos 33

4.2. Recursos de Terceros 34

4.2.1. *Hugging Face* 34

4.2.2. Proyectos Independientes 35

4.2.3. *spaCy* y *NLTK* 36

4.2.4. *Wikidata* 38

4.2.5. *Gradio* 39

4.2.6. Limitaciones y Consideraciones 39

4.3. Módulo de Preprocesamiento 41

4.3.1. Translator 42

4.3.2. Summarizer 42

4.3.3. StraightTextStats 43

4.4. Módulo de Extracción de Conceptos Claves 43

4.5. Módulo de Detección de Entidades 43

4.5.1. Desambiguación de Entidades 43

4.6. Módulo de Creación de Ilustraciones 44

4.7. Desarrollo de la Interfaz 44

5. Evaluación y Validación de Resultados **45**

5.1. Ejemplo Guiado 45

5.1.1. Inicialización del Sistema 45

5.1.2. Utilización del Sistema 46

5.2. Validación con Usuarios 53

5.2.1. Metodología 53

5.2.2. Resultados 54

6. Discusión **56**

6.1. Implicancias 56

6.2. Limitaciones 57

6.3. Trabajo Futuro 57

7. Conclusión **58**

Bibliografía **61**

Índice de Ilustraciones

1.1. Ejemplar de arte rupestre representando una escena de caza [9]. 2

1.2. Ejemplo de imágenes generadas por modelos de difusión [2]. 4

2.1. Representación de un Perceptrón Multicapa [13]. 9

2.2. Representación de una red neuronal donde la función de activación corresponde a la función escalón unitario [13]. 9

2.3. Flujo del proceso de entrenamiento de una red neuronal [26]. 10

2.4. Etapas del Procesamiento del Lenguaje Natural. 13

2.5. Pasos del *pipeline* para el Procesamiento del Lenguaje Natural [31]. 14

2.6. Ejemplo de *embedding* generado a partir de una entrada en *one-hot encoding* [5]. 16

2.7. Modelo de arquitectura de un *Transformer* [32]. 18

2.8. Comparativa entre Modelos Generativos [11]. 23

2.9. Representación del comportamiento de un modelo de difusión [19]. 24

2.10. The Dreamcatcher: una herramienta visual interactiva que explora el vínculo entre los sueños y la vida de las personas [3]. 25

3.1. Arquitectura del *pipeline* de procesamiento. 30

4.1. Representación del *pipeline* de procesamiento de *spaCy* [29]. 36

5.1. Primera vista a la aplicación. 46

5.2. Nube de palabras. 47

5.3. Raíces de palabras más frecuentes. 47

5.4. Interfaz de la pestaña “Extractor de Conceptos Clave”. 48
5.5. Conceptos Clave. 48
5.6. Interfaz de la pestaña “Extractor de Entidades”. 49
5.7. Ejemplo de Entidad Detectada. 50
5.8. Ejemplo de una segunda Entidad Detectada. 51
5.9. Interfaz de la pestaña “Ilustraciones del texto”. 52
5.10. Dua Lipa en el Cerro San Cristóbal. 52

Capítulo 1

Introducción

1.1. Contexto

La literatura y los textos no literarios han provocado un impacto significativo y transformador sobre diversas sociedades, desempeñando un papel fundamental a lo largo de la historia de la raza humana. Estos fenómenos se han manifestado y han estado presentes en diversas civilizaciones, adquiriendo una importancia e influencia considerable en diferentes aspectos sociales y períodos históricos, dejando huellas firmes que han forjado el rumbo que hemos tomado como humanidad.

Al analizar un poco la historia, es fácil notar que desde la época de las antiguas civilizaciones como Mesopotamia, Egipto, Grecia y Roma, la literatura desempeñó un rol crucial en la preservación de la cultura y las tradiciones. Gracias a los poemas épicos, como "La Ilíada" y "La Odisea", las civilizaciones fueron capaces de preservar y transmitir los ideales, valores y creencias de sus sociedades, así como también las hazañas de sus héroes. Incluso se utilizaron textos escritos para el registro de leyes, tratados y conocimientos científicos, lo cual contribuyó enormemente al desarrollo de estas culturas.

Por otro lado, durante la Edad Media la literatura fue fuertemente influenciada por la religión, y fue utilizada como una herramienta para difundir la fé y la moralidad. Los escritos religiosos, como la Biblia, moldearon las creencias y las prácticas de la sociedad feudal, incluso llegando a influir en las artes y arquitectura de la época.

Ya entrando en la Edad Moderna, la literatura se convirtió en un medio para la exploración de nuevas ideas y la difusión del conocimiento. Además, con los cambios de paradigma y las nuevas tecnologías surgidas de la revolución industrial, los textos y la escritura jugaron un papel crucial en el cambio social y político, siendo utilizadas como una forma de crítica social, capaz de resaltar las desigualdades e injusticias de la época y de cuestionar el poder establecido.

Este patrón se puede observar a lo largo de diversos periodos históricos de la raza humana, teniendo momentos con mayor o menor impacto, pero siempre manteniéndose como un elemento presente, incluso hasta el día de hoy.

En la actualidad, la literatura continúa ejerciendo una influencia significativa en la sociedad de diversas formas. Se emplea como una herramienta de aprendizaje que brinda a los lectores acceso a distintas visiones de mundo y perspectivas, ampliando y enriqueciendo la comprensión del entorno que los rodea. Asimismo, fomenta la creatividad y la imaginación, actuando como una puerta de entrada hacia mundos de fantasía que transportan a los lectores en un viaje mágico. Además, la literatura posee una capacidad inmensa para transmitir experiencias humanas y evocar emociones en los lectores. Puede ofrecer entretenimiento, inspiración e incluso provocar reflexiones profundas en aquellos que se sumergen en sus páginas.

Cabe mencionar que un porcentaje del éxito de la literatura está estrechamente relacionado con la utilización de ilustraciones para representar y transmitir de manera más efectiva lo que se desea comunicar. Esta combinación ha potenciado la capacidad de comunicación y transmisión de mensajes. Incluso en épocas como la prehistoria, donde no existían textos literarios en el sentido tradicional o la escritura en sí, las ilustraciones desempeñaron un papel fundamental en la transmisión de información e historias. Los seres humanos utilizaban pinturas rupestres y grabados en cuevas para representar escenas de caza, rituales, animales y otros aspectos de su vida cotidiana. Estas ilustraciones se convirtieron en una forma temprana de comunicación visual, permitiendo a las personas transmitir sus experiencias, conocimientos y narrativas de manera detallada y eficaz (Ver Figura 1.1).

Figura 1.1: Ejemplar de arte rupestre representando una escena de caza [9].

Las ilustraciones complementan y enriquecen al texto escrito, creando atmósferas, aportando detalles visuales, y ayudando a visualizar eventos, acciones, lugares y entidades descritas en el relato. Por medio de las ilustraciones, la literatura y los textos en general, se vuelven más accesibles y atractivos, especialmente para los lectores que prefieren una experiencia visual más inmersiva.

No obstante, no todo lo relacionado con el mundo de la escritura son aspectos positivos. En él se presentan una serie de desafíos y complicaciones que hay que tener en consideración. Afrontar un proceso de redacción o un proceso creativo no es sencillo, y se requiere una considerable inversión de tiempo, esfuerzo y dedicación para crear y redactar una obra que logre transmitir de manera fiel las ideas, historias o experiencias que se desean comunicar.

Si bien el proceso de escritura es un mundo que varía de escritor en escritor, uno de los desafíos comunes a los que estos se enfrentan son los bloqueos creativos. Durante estos bloqueos, los escritores experimentan una falta de inspiración e ideas atractivas, así como dificultades para encontrar las palabras adecuadas y evocadoras que les permite expresarse de manera precisa y efectiva. Esta situación puede resultar frustrante y dificultar el progreso en la escritura. Pero incluso atravesado este proceso, las complicaciones siguen, es importante mantener la coherencia y una buena estructura en el relato, evitando abusar de recursos literarios o de repetición de palabras que generen un círculo vicioso en torno a una misma idea. Además, es fundamental destacar la importancia de la edición y revisión meticulosa, la cual permite la corrección de errores gramaticales, el mejoramiento de la calidad del lenguaje y en general factores esenciales que permiten obtener un resultado final satisfactorio.

1.2. Motivación

Algunas soluciones y técnicas utilizadas por escritores para sobreponerse ante las dificultades en los procesos creativos incluyen tomar periodos de descanso, buscar fuentes de inspiración, mejorar y ordenar sus zonas de trabajo, entre otras. En cuanto a la coherencia y estructura, suelen utilizar notas, diagramas, mapas conceptuales y en general aplicaciones que facilitan la organización de la información. Por el lado de la edición, los escritores cuentan con el apoyo de tecnologías que incorporan correctores gramaticales, y realizan sesiones iterativas a lo largo del proceso editorial para pulir su obra y transmitir de la mejor manera posible lo que desean comunicar. Dicho esto, la creación de una obra literaria es un proceso agotador que requiere una inmensa cantidad de esfuerzo, dedicación y persistencia, donde la implementación e integración de algunas técnicas y tecnologías al flujo de trabajo pueden facilitar en cierta medida este proceso.

No obstante, el beneficio que estos factores y tecnologías aportan a los escritores podría ser mucho mayor. En la actualidad, las aplicaciones y técnicas de inteligencia artificial (AI) son utilizadas en múltiples áreas. Industrias como la automotriz, la salud, la educación, entre otras, han sido objeto de investigación y se han visto beneficiadas por las capacidades de la inteligencia artificial. Incluso conceptos como el lenguaje no se quedan atrás en esta categoría. De hecho, existe todo un campo dedicado al estudio del procesamiento del lenguaje natural (NLP), donde se pueden identificar dos áreas principales: el entendimiento del lenguaje natural y la generación de lenguaje natural. Las técnicas relacionadas con la primera rama permiten que las máquinas comprendan el lenguaje natural y sean capaces de extraer palabras clave, conceptos, entidades e incluso emociones en diferentes representaciones del lenguaje. Mientras que la segunda rama permite que las máquinas generen lenguaje natural de manera coherente y comprensible.

Dicho esto, una de las representaciones más comunes del lenguaje natural es el lenguaje escrito. Por este motivo, el procesamiento de texto ha sido un tópico recurrente en NLP, y se han desarrollado múltiples técnicas, modelos y enfoques para realizar análisis a mayor escala y de manera más efectiva. En particular, las técnicas relacionadas con el aprendizaje profundo o *deep learning* han experimentado un crecimiento significativo en la última década[1].

[1]https://www.forbes.com/sites/louisaxu/2021/12/01/a-golden-age-for-natural-language/

Este crecimiento se debe principalmente a los avances teóricos en computación, al mejoramiento del rendimiento de los recursos computacionales y al aumento de la accesibilidad a fuentes de datos e información. Esto ha permitido entrenar y generar modelos más sofisticados, los cuales utilizan técnicas y arquitecturas bastante potentes. Estos modelos son conocidos como "*Large Language Models*" (desde ahora *LLMs*), y son modelos multitarea no supervisados entrenados con grandes volúmenes de datos de texto, siendo capaces de realizar variadas tareas con buenos resultados. Además, pueden ser especializados en áreas específicas para obtener un mejor rendimiento en tareas comunes de *NLP*, como el análisis de dependencias, el reconocimiento de entidades, el análisis de sentimientos, la generación de texto, la traducción automática, entre otras. La capacidad de los *LLMs* para adaptarse a diferentes tareas y disciplinas, junto con sus excelentes resultados, ha contribuido en gran medida a su popularidad actual, dominando hoy en día el estado del arte en *NLP* y convirtiéndose en un tópico que está en constante evolución.

Otra tendencia que se ha manifestado estos últimos años en el mundo de la inteligencia artificial son los modelos generativos. Estos modelos son utilizados para generar nuevos datos, como imágenes, texto, música o incluso videos, que se asemejan a los datos de entrenamiento proporcionados previamente. Una de las características de los modelos generativos es que son capaces de aprender patrones y estructuras subyacentes en los datos de entrenamiento y luego utilizar este conocimiento para generar nuevas muestras que son similares en estilo, contenido y distribución a los datos originales. Los modelos de difusión son un tipo de modelo generativo de datos que ha conseguido resultados asombrosos en aplicaciones de síntesis de imágenes, lo cual ha impulsado significativamente su utilización en la industria creativa y de diseño. Ejemplo de esto son las imágenes de la figura 1.2 generadas por modelos de difusión.

Figura 1.2: Ejemplo de imágenes generadas por modelos de difusión [2].

Es así que ante esta serie de desafíos y nuevas tecnologías surge la oportunidad de ayudar a los escritores a superar estos obstáculos mediante el aprovechamiento de técnicas de *NLP*, *LLMs* y modelos generativos. La idea es verificar la factibilidad, y en caso de probarse, proporcionar a los escritores y gente interesada una herramienta innovadora, accesible e intuitiva que les permita potenciar su creatividad y apoyarlos en su escritura de manera más eficiente, ayudándolos a superar algunas dificultades y perfeccionar su proceso creativo.

La escritura efectiva y evocadora desempeña un papel fundamental en diversos campos, como la literatura, el periodismo, el *marketing* y la comunicación en general. Los escritores buscan constantemente formas de mejorar su escritura y destacarse en un entorno cada vez más competitivo. La capacidad de expresar ideas de manera precisa y cautivadora puede marcar la diferencia en el impacto y la recepción de su trabajo.

Por lo tanto, al proponer una herramienta que tome en consideración algunas de las complicaciones comunes que enfrentan los escritores al sumergirse en procesos creativos, el objetivo es proporcionar apoyo y recursos que les permitan enfrentar estos desafíos de manera más eficaz, brindándoles una ventaja en la generación de contenido. Con esto en mente, se intentará desarrollar un sistema que facilite la identificación de palabras clave, el reconocimiento de las entidades y las relaciones entre los elementos del texto, además de brindar apoyo ilustrativo al relato que se esté analizando. Con esta aplicación, se busca potenciar la industria creativa al proporcionar a los escritores las herramientas necesarias para perfeccionar su trabajo y alcanzar nuevos niveles de excelencia en sus creaciones.

1.3. Objetivos

1.3.1. Objetivo General

El objetivo de este proyecto consiste en investigar la factibilidad, y posteriormente diseñar y desarrollar un *MVP* de aplicación en la cual los usuarios puedan ingresar fragmentos de texto para que sean procesados, analizados y caracterizados por medio de un *pipeline* de procesamiento capaz de generar contenido adicional que sea relevante y útil para estos, sirviendo como fuente de inspiración y ayudándolos eventualmente a superar problemas relacionados con los procesos creativos y de redacción.

Entre las tareas relacionadas con este objetivo, se incluyen la detección e identificación de patrones, palabras recurrentes, conceptos clave, entidades e interacciones presentes en los relatos ingresados. Además, la aplicación permitirá visualizar representaciones sintácticas y semánticas de los textos, donde las ilustraciones semánticas serán generadas por modelos de difusión mediante la entrega de un conjunto de palabras que estimulen la generación, o en otras palabras, *prompts*[6] creadas de manera automática a partir de la estructura de los relatos.

1.3.2. Objetivos Específicos

1. Aplicar técnicas de procesamiento de texto y *NLP* en textos ingresados por usuarios.
2. Extraer patrones, palabras recurrentes, conceptos clave, entidades e interacciones que se detecten, y compararlas y complementarlas con fuentes de datos como Wikidata.
3. Generar *prompts* adecuadas y de manera automática a partir del análisis y procesamiento de uno o más relatos ingresados por los usuarios, las cuales representen y conserven de buena manera la idea original de los textos.

4. Generar visualizaciones a nivel individual, tanto de la estructura sintáctica del texto vista a través de las entidades, conceptos, interacciones, como representaciones ilustradas a través de modelos de difusión por medio de las *prompts* previamente generadas.

5. Integrar y construir un único *pipeline* de procesamiento que incorpore y aplique las técnicas de *NLP*, *LLMs* y modelos de difusión previamente utilizados.

6. Modularizar y documentar la herramienta de manera que sea un trabajo autocontenido y explicativo, otorgando la posibilidad de que pueda ser continuada y mejorada como un trabajo a futuro.

7. Disponibilizar la aplicación en forma de *MVP* para que pueda ser utilizada por usuarios.

1.4. Metodología

Para lograr los objetivos establecidos, se adoptó una metodología estructurada que permitió llevar a cabo el diseño, desarrollo e integración de la herramienta. A continuación, se detallan las etapas principales que se siguieron durante el proceso:

1. **Investigación:** En esta etapa se llevó a cabo una investigación exhaustiva sobre el estado del arte en técnicas de *NLP*, *LLMs* y modelos generativos, incluyendo su funcionamiento, fortalezas y debilidades. Se analizaron librerías y modelos utilizados en el procesamiento de texto, capaces de realizar tareas como extracción de palabras clave, reconocimiento de entidades y generación de resúmenes. Adicionalmente, se analizaron algunas pruebas de concepto con modelos de difusión capaces de generar imágenes. Todo esto con el objetivo de identificar las herramientas más adecuadas para implementar un *MVP* de aplicación.

2. **Diseño e Implementación de la Solución:** En esta etapa se llevó a cabo el trabajo relacionado con la planeación de la arquitectura y el desarrollo de la solución para ayudar a los escritores a superar bloqueos creativos y mejorar sus escritos. Fue la fase más demandante del proceso, en la cual se realizó toda la programación necesaria para el desarrollo de un sistema de procesamiento de texto que permita a los usuarios ingresar párrafos y obtener análisis detallados sobre ellos. Para lograr esto, se integraron los modelos y técnicas previamente investigadas en la construcción de un *pipeline* de procesamiento, y se desarrolló una interfaz amigable para que los usuarios pudieran interactuar con la herramienta de manera intuitiva.

3. **Disponibilidad de la Aplicación:** Una vez desarrollada e integrada la herramienta, se procedió a planificar e implementar su disponibilidad para los usuarios. Se tomaron en consideración varios formatos y opciones de uso, con el objetivo de entregar mayor flexibilidad y control a los usuarios. Además, se realizaron pruebas de integración para asegurar el correcto funcionamiento de los distintos métodos de ejecución y entornos desplegados.

4. **Evaluación y Validación de Resultados:** En esta etapa, se evaluaron y validaron los resultados obtenidos por la herramienta. Se realizaron pruebas utilizando diferentes ejemplos de textos y se midió el desempeño de la aplicación y la calidad de los resultados por medio de pruebas realizadas a un conjunto pequeño y acotado de usuarios, los cuales posteriormente respondieron una encuesta con respecto a su nivel de satisfacción con la herramienta.

5. **Modularización, Documentación y Conclusiones:** En esta etapa final, se modularizó el código de la herramienta para facilitar su mantenimiento y futuras actualizaciones. Se generó una documentación completa que incluyó una descripción detallada de las funcionalidades de la herramienta, las técnicas y modelos de *NLP* utilizados y las instrucciones de uso para los usuarios. Finalmente, se concluyó el trabajo destacando los logros alcanzados, las limitaciones identificadas y las posibles futuras mejoras para la aplicación desarrollada.

1.5. Estructura del Documento

El presente documento, que expone el proyecto realizado a modo de trabajo de título, está dividido en 7 capítulos.

En el capítulo actual, de introducción, se establece el contexto, motivación y objetivos del trabajo realizado durante el desarrollo de la herramienta. Mientras que el resto del documento es estructurado de la siguiente manera:

Capítulo 2 - Marco Teórico: Se presentan y describen los elementos conceptuales necesarios para la comprensión de técnicas de *NLP*, *LLMs* y modelos generativos utilizados en el proyecto. Además, se mencionan otros elementos y plataformas que jugaron un rol crucial en el desarrollo del trabajo.

Capítulo 3 - Diseño de la Solución: Se plantea el diseño y la arquitectura de la aplicación desarrollada utilizando los conocimientos presentados en el Capítulo 2. En particular, se mencionan algunos requisitos previos para la construcción de la herramienta, se describe su arquitectura general y se presentan módulos que componen la aplicación.

Capítulo 4 - Implementación de la Solución: Se describe el desarrollo, construcción e integración de la aplicación en términos más técnicos. Se detalla cada módulo con mayor profundidad y con un nivel más bajo de abstracción, indicando más claramente cómo todos los módulos interactúan entre sí y mencionando los recursos de terceros que son utilizados en la herramienta.

Capítulo 5 - Evaluación y Validación de Resultados: Se evalúa y se pone a prueba el rendimiento y los resultados de la aplicación por medio de un ejemplo guiado que sirve como apoyo para comprender su utilización. Además, se explica en que consistió el proceso de validación realizado por los usuarios, se presentan los resultados obtenidos y los niveles de satisfacción expresados por estos.

Capítulo 6 - Discusión: Se discuten los resultados obtenidos por la herramienta, se identifican algunas de sus limitaciones y se proponen posibles trabajos futuros que se puedan realizar sobre la aplicación desarrollada.

Capítulo 7 - Conclusión: Se presentan las conclusiones del trabajo realizado, resumiendo los logros más importantes y las dificultades encontradas durante el desarrollo del proyecto, además de destacar la contribución de la herramienta.

Capítulo 2

Marco Teórico

Previamente, a modo de introducción y contextualización, se presentaron y mencionaron algunos elementos conceptuales que guardan relación con el trabajo realizado. Sin embargo, estas menciones fueron superficiales y sirvieron únicamente como contexto. Por lo cual, en esta sección se profundizará en los conceptos con el fin de adquirir un conocimiento adecuado para comprender las tecnologías, modelos y herramientas utilizadas en el proyecto.

2.1. Antecedentes

Las redes neuronales artificiales y el *deep learning* son dos conceptos fundamentales en el campo de la inteligencia artificial y el aprendizaje automático. Los descubrimientos y tecnologías asociadas a ellos han revolucionado muchos campos de estudio y aplicaciones prácticas, demostrando un gran potencial para resolver problemas complejos y realizar tareas que antes parecían imposibles.

Por ejemplo, su impacto se ha extendido a ámbitos como la visión por computadora y el entendimiento del lenguaje humano por medio de la comprensión de texto o reconocimiento de voz, lo que ha desencadenado en aplicaciones prácticas como la conducción autónoma y la creación de *Chatbots* respectivamente. Las capacidades para procesar grandes volúmenes de datos y extraer patrones complejos ha permitido el desarrollo de sistemas más inteligentes y eficientes. Además, estas técnicas han mejorado la precisión y la velocidad de muchos procesos, lo que ha llevado a una mayor automatización y optimización en numerosas industrias.

A medida que se continúa investigando y refinando estos métodos, se espera que su impacto en la sociedad y la tecnología siga creciendo. Se vislumbran nuevas soluciones y posibilidades en el ámbito de la inteligencia artificial, lo que promete un futuro lleno de avances y oportunidades.

Dicho esto, y teniendo en consideración que estos conceptos son las bases de las tecnologías usadas en el proyecto, se procederá a ofrecer una explicación más detallada sobre estos términos.

2.1.1. Redes Neuronales Artificiales

Las redes neuronales artificiales son modelos computacionales inspirados en el funcionamiento del cerebro humano. Están compuestas por un conjunto de muchas unidades interconectadas entre sí llamadas neuronas artificiales o perceptrones, que a su vez se organizan en capas, formando una arquitectura conocida como perceptrón multicapa (Ver Figura 2.1).

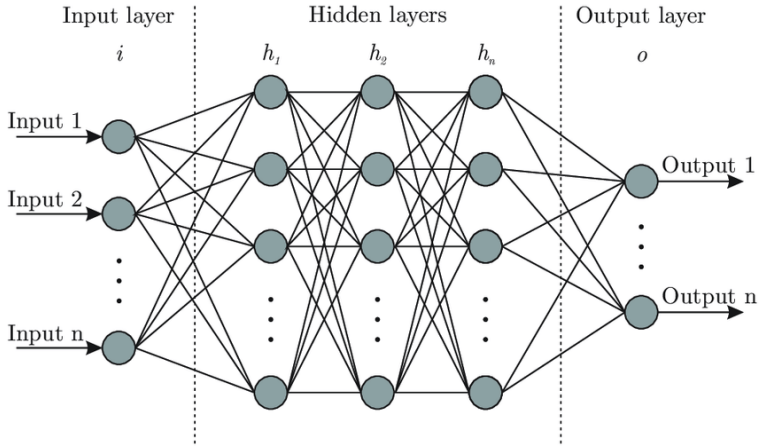

Figura 2.1: Representación de un Perceptrón Multicapa [13].

El funcionamiento de estas redes se basa en una serie de pasos que son repetidos iterativamente capa por capa hasta llegar a la capa final. Cada neurona artificial procesa una señal o *input* numérico recibido como entrada, aplicando una función de activación no lineal a la combinación lineal de las entradas, y transmitiendo el resultado o salida producida a las neuronas de la capa siguiente a través de conexiones ponderadas.

Estas conexiones ponderadas, también conocidas como pesos (*weights*), representan la fuerza e influencia de cada neurona en la generación de la salida final de la red. Además de los pesos, las redes neuronales también incluyen un parámetro adicional llamado sesgo (*bias*), que se suma a la combinación lineal de las entradas ponderadas antes de pasar por la función de activación. El sesgo permite ajustar el punto de partida de la activación de cada neurona, lo que puede ser útil para modelar relaciones no lineales más complejas. La figura 2.2 representa una red neuronal, mostrando todos los parámetros previamente discutidos y sus influencias en el *output* de la red.

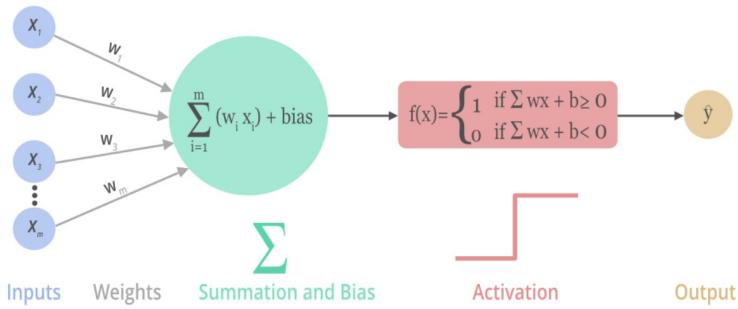

Figura 2.2: Representación de una red neuronal donde la función de activación corresponde a la función escalón unitario [13].

El objetivo de las redes neuronales artificiales es aprender y mejorar su comportamiento en una tarea basándose en ejemplos, sin conocimiento previo en la tarea abordada ni una programación específica para poder realizarla. El hecho de que cada neurona artificial procese *inputs* numéricos aplicando una función de activación no lineal a la combinación lineal de estos, agrega no linealidades a la red neuronal, lo cual es esencial para que la red pueda aprender relaciones y patrones complejos de los datos con los cuales se está trabajando.

Por otro lado, el hecho de que las salidas de las neuronas de una capa sean propagadas a través de conexiones ponderadas hacia las neuronas de la capa siguiente, donde el proceso de combinación lineal, activación y transmisión de la salida es repetido capa por capa hasta llegar a la capa de salida, constituye el proceso fundamental que permite a una red neuronal artificial aprender y realizar tareas complejas como reconocimiento de patrones, clasificación, predicción, entre muchas otras aplicaciones más en el campo del aprendizaje automático.

El poder de esta serie de pasos que se repiten iterativamente capa a capa, conformando el proceso fundamental que moldea la naturaleza y comportamiento de una red neuronal, radica en que se le otorga a la red neuronal la capacidad de ajustar los pesos de sus conexiones. Provocando que la red pueda aprender a mapear una serie de entradas a una serie de salidas deseadas. Esto se logra por medio de un proceso llamado entrenamiento, donde se presentan a la red ejemplos de entrada junto con las salidas esperadas, y la red ajusta iterativamente los pesos de sus conexiones para minimizar la diferencia entre las salidas producidas y las salidas esperadas.

El entrenamiento de una red neuronal se realiza mediante un proceso iterativo conocido como *Algoritmo de Retropropagación del Error* [25]. Este algoritmo ajusta los pesos de las conexiones en función de la diferencia entre la salida predicha por la red y la salida deseada, propagando el error hacia atrás a través de las capas. De esta manera, la red aprende a mejorar sus predicciones y a ajustar sus conexiones para minimizar el error.

A medida que la red se entrena con más ejemplos, los pesos de las conexiones se ajustan para capturar las relaciones y patrones presentes en los datos de entrenamiento. La no linealidad introducida por las funciones de activación permite que la red pueda modelar relaciones complejas y no lineales entre las entradas y las salidas. En la figura 2.3, se presenta un diagrama que grafica todo el flujo que constituye el proceso de entrenamiento de una red.

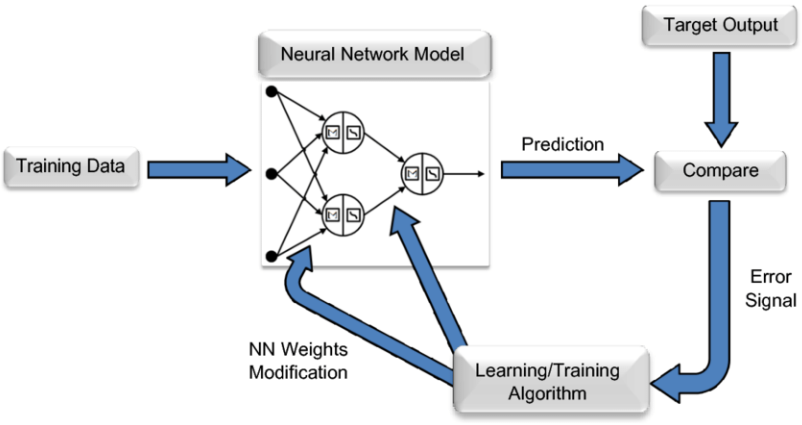

Figura 2.3: Flujo del proceso de entrenamiento de una red neuronal [26].

Una vez que la red ha sido entrenada, se espera que pueda generalizar su aprendizaje a nuevos datos que no ha visto durante el entrenamiento. Esto significa que la red podrá producir salidas adecuadas, incluso para datos de entrada con los cuales no ha interactuado previamente, sin ningún conocimiento previo en ellos y ni una programación específica en la tarea que se está realizando.

Aunque es cierto que en su esencia las redes neuronales trabajan con números, ya que los datos de entrada y salida se representan numéricamente, es importante destacar que su utilidad no se limita únicamente al procesamiento de datos numéricos. Gracias a su capacidad para aprender patrones y relaciones complejas, las redes neuronales también pueden ser aplicadas a problemas de naturaleza no numérica, como el procesamiento de texto, imágenes, audio y video. Al utilizar técnicas de representación adecuadas, es posible convertir estos datos en formatos que las redes neuronales puedan entender y analizar. Esto amplía significativamente el alcance de las aplicaciones de las redes neuronales, permitiendo abordar una amplia gama de desafíos en diferentes dominios. Esta flexibilidad y capacidad de adaptación es lo que convierte a las redes neuronales artificiales en herramientas poderosas en el campo del aprendizaje automático y la inteligencia artificial.

2.1.2. Deep Learning

El aprendizaje profundo o *deep learning* (*DL*) es una rama del aprendizaje de máquinas o *machine learning* (*ML*) que se basa en el uso de redes neuronales artificiales con múltiples capas ocultas para el procesamiento de la información. Las principales diferencias entre las redes neuronales tradicionales y las redes neuronales profundas o *Deep Neural Networks* (*DNN*) son las siguientes:

1. **Arquitectura de capas:** Mientras que las redes neuronales tradicionales generalmente constan de una o dos capas ocultas, las *DNN* están compuestas por múltiples capas ocultas. Cada capa se encarga de extraer características específicas y representaciones más abstractas y complejas de los datos de entrada, lo que permite un procesamiento más profundo y sofisticado. Esta arquitectura más profunda también les otorga una mayor capacidad para capturar patrones subyacentes en los datos y aplicar ese conocimiento a nuevas muestras, lo que se traduce en una mayor capacidad de generalización.
2. **Aprendizaje automático de representaciones:** A diferencia de las redes neuronales tradicionales donde, independientemente de la tarea o problema abordado, los datos se preprocesan antes de ser entregados a la red. En una *DNN* los datos son entregados en su estado más puro, y es la red la que aprende la mejor manera de representar la data y las características relevantes se aprenden automáticamente durante el entrenamiento, no existiendo un preprocesamiento en los datos.
3. **Requerimientos computacionales:** Debido a su arquitectura más compleja y profunda, las *DNN* suelen requerir mayores recursos computacionales, tanto en términos de capacidad de procesamiento como de memoria. El entrenamiento de estas redes puede ser computacionalmente intensivo y a menudo se benefician del uso de unidades de procesamiento gráfico (GPU) o unidades de procesamiento tensorial (TPU) para acelerar el procesamiento paralelo.

Finalmente, son estas capas ocultas adicionales que constituyen una *DNN* las que permiten que las redes aprendan representaciones más sofisticadas y complejas a partir de los datos de entrada. A medida que los datos fluyen a través de estas capas, cada una de ellas se encarga de aprender gradualmente características más abstractas y de más alto nivel. Esta estrategia ha demostrado ser particularmente efectiva en la resolución de problemas desafiantes y la obtención de resultados precisos para tareas como el reconocimiento de objetos en imágenes y el procesamiento del lenguaje natural.

Por otro lado, el éxito de las *DNN* se ha visto impulsado por diversos factores. En primer lugar, los avances en el *hardware* de computación, como las unidades de procesamiento gráfico (GPU) y los sistemas de computación distribuida, han permitido acelerar tanto el entrenamiento como la inferencia de estas redes neuronales profundas de mayor tamaño y complejidad, reduciendo significativamente los tiempos de procesamiento. Además, el aumento en la disponibilidad de grandes conjuntos de datos ha brindado la oportunidad de entrenar modelos más precisos y generalizados.

Todo esto apunta hacia un futuro fascinante, en el que las *DNN* desempeñarán un papel protagonista. Con su capacidad para aprender representaciones complejas y abstractas, respaldadas por avances en *hardware* y conjuntos de datos masivos, las *DNN* nos brindan un camino hacia soluciones poderosas y precisas en una amplia gama de aplicaciones. Este deslumbrante horizonte promete transformar radicalmente campos como la visión por computadora, el procesamiento del lenguaje natural y muchas otras disciplinas, abriendo nuevas posibilidades y desafiando los límites de lo que se puede lograr con inteligencia artificial.

2.2. Procesamiento del Lenguaje Natural

El procesamiento del lenguaje natural (*NLP*) es una rama de la inteligencia artificial que se enfoca en la interacción entre las computadoras y el lenguaje humano. El objetivo principal del *NLP* es permitir que las máquinas analicen, comprendan y generen texto o lenguaje humano de manera efectiva. La premisa fundamental es que si las máquinas son capaces de comprender y generar el lenguaje humano con un alto nivel de entendimiento y coherencia, podrán llevar a cabo tareas a través de formatos de comunicación efectivos, como el lenguaje escrito o hablado.

A medida que la cantidad de datos de texto generados por humanos continúa creciendo exponencialmente, y que la presencia de computadoras y dispositivos similares como *smartphones* y *tablets* se vuelve más común, la necesidad de una mejor y más rápida comunicación entre humanos y máquinas se vuelve imperativa. Es debido a esto que la principal preocupación de este campo es encontrar los mejores métodos y técnicas para traducir estas expresiones del lenguaje en datos que sean comprensibles por las máquinas [18].

Dicho esto, el *NLP* se ha vuelto cada vez más importante, existiendo en la actualidad diversos enfoques y técnicas utilizadas para analizar y comprender el lenguaje. Muchas de estas basan sus fundamentos en el aprovechamiento del poder del *deep learning*, lo que se ha traducido en un avance significativo en la capacidad de las máquinas para comprender y generar lenguaje humano.

2.2.1. Áreas y Fundamentos del *NLP*

Como se mencionó en la motivación del proyecto, en el campo del *NLP* es posible identificar dos áreas principales: El entendimiento del lenguaje natural o *Natural Language Understanding* (*NLU*) y la generación del lenguaje natural o *Natural Language Generation* (*NLG*).

Entendimiento del Lenguaje Natural (*NLU*:) Esta área se centra en desarrollar algoritmos y técnicas que permiten a las máquinas comprender el lenguaje humano en diversas formas. El objetivo es capacitar a las computadoras para interpretar y extraer significado de textos escritos o hablados, como documentos, mensajes, notas de voz, etc.

Generación del Lenguaje Natural (*NLG*): Esta área se enfoca en desarrollar sistemas capaces de producir texto legible y coherente en lenguaje humano como respuesta a diferentes *inputs* o tareas. Los *chatbots*, las herramientas de traducción automática, la generación automática de resúmenes de texto, o los asistentes de voz impulsados por inteligencia artificial son algunas de las aplicaciones de *NLG*.

Estas dos áreas son fundamentales en el desarrollo de aplicaciones de procesamiento del lenguaje natural y juegan un papel crucial en la mejora de la interacción entre humanos y máquinas, así como en la automatización de tareas relacionadas con el lenguaje. Debido a que en su esencia la comunicación es la acción de intercambiar información, lo cual se logra a través del entendimiento y generación de nuevo contenido, el avance y progreso en cualquiera de estas dos áreas beneficia enormemente a las aplicaciones prácticas que permiten la interacción entre humanos y máquinas.

Debido a la naturaleza del proyecto, nos centraremos en técnicas y modelos para el procesamiento de texto que tengan relación con ambas áreas. Ya que, se busca comprender los relatos de los usuarios, pero también se busca proponer análisis y contenido que aporten valor para estos. Sin embargo, antes de comenzar a hablar sobre estas técnicas, es importante aclarar algunos conceptos que son mencionados frecuentemente en este dominio.

En el caso general, para realizar cualquier tarea de *NLP* que involucre texto, es necesario atravesar una serie de etapas que conforman el proceso estándar mediante el cual el lenguaje natural puede ser comprendido por las máquinas (Ver Figura 2.4).

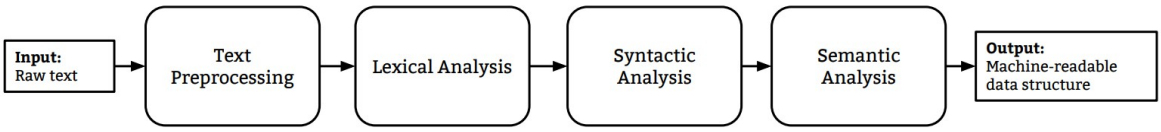

Figura 2.4: Etapas del Procesamiento del Lenguaje Natural.

Este proceso consta de distintos pasos y análisis dentro de sus etapas. Y es gracias a ellos, y a las características que estos extraen, que una entrada en texto plano puede ser procesada, descompuesta y transformada en una estructura de datos comprensible para una máquina, pero que a su vez mantenga el sentido, coherencia y esencia del texto que está siendo analizado.

En particular, esta serie de pasos fundamentales que trabajan de manera secuencial para analizar y comprender el texto, conforman un *pipeline* de procesamiento del lenguaje natural. A continuación, en la figura 2.5 se presenta una ilustración con los principales pasos que componen este *pipeline* y se brinda una explicación sobre en qué consisten:

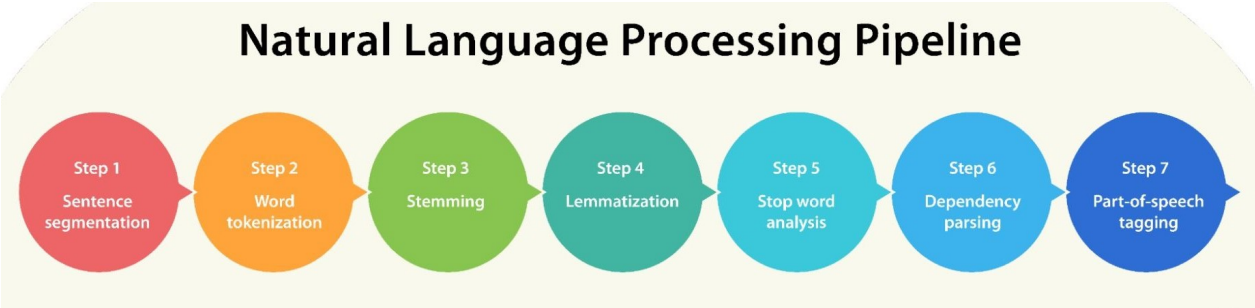

Figura 2.5: Pasos del *pipeline* para el Procesamiento del Lenguaje Natural [31].

Sentence Segmentation: El proceso de segmentación de oraciones es un procedimiento que se utiliza para separar párrafos en oraciones individuales. Esto es importante porque muchas tareas de *NLP* se realizan a nivel de oración, y la segmentación adecuada es fundamental para un análisis preciso del texto. Este paso presenta sus propios desafíos, por ejemplo caracteres como los puntos a menudo son utilizados para marcar el final de una oración, pero no ocurre siempre así, ya que también son utilizados para denotar abreviaturas y números decimales.

Word Tokenization: Cada oración se divide en unidades más pequeñas llamadas *tokens*. Los *tokens* pueden ser caracteres, subpalabras o palabras individuales. La tokenización es un paso esencial para que las palabras individuales puedan ser procesadas y analizadas por los algoritmos de *NLP*.

Stemming: Proceso heurístico que consiste en la eliminación de sufijos y prefijos de palabras para dejarlos en un invariante forma canónica o "raíz". El resultado del stemming no siempre es una palabra real, ya que la raíz puede no tener significado por sí misma, generando palabras truncadas o incompletas. Por ejemplo, las palabras "corriendo", "corre", "correrá" se reducirían a la raíz "corr-" mediante el proceso de stemming. Esto ayuda a simplificar las palabras y agrupar diferentes formas gramaticales bajo una única forma base.

Lemmatization: La lematización es un proceso algorítmico análogo y con el mismo propósito que el *stemming*, pero se logra a través de un conjunto de pasos más riguroso que incorporan el análisis morfológico de cada palabra. Es más complejo y preciso, logrando reducir las palabras a su forma canónica o lema, que es la forma base con significado léxico real. El resultado de la lematización siempre será una palabra real y válida en el idioma. Por ejemplo, si se aplica la lematización a las palabras 'corriendo", "corre" y "correrá", cada una se reducirá a su lema correspondiente: "correr".

Stop Word Analysis: Consiste en la eliminación de palabras vacías o *stopwords* del texto. Las palabras vacías son términos muy comunes en el lenguaje (como artículos, pronombres, preposiciones, etc.) que generalmente no aportan un significado relevante al análisis y pueden ser ignoradas para reducir la dimensionalidad del texto.

Dependency Parsing: El análisis de dependencias consiste en analizar la estructura gramatical de una oración y representar las relaciones de dependencia entre las palabras. Generalmente, se utiliza una estructura de grafo dirigido conocida como árbol de dependencias para representar las dependencias gramaticales de las oraciones.

Part-of-Speech (POS) Tagging: Se asigna una etiqueta gramatical (como adjetivo, sustantivo, verbo, adverbio, preposición, etc.) a cada palabra en el texto. El etiquetado gramatical es útil para entender la estructura gramatical de las oraciones y es crucial para muchas tareas de *NLP*, como el análisis sintáctico y la extracción de información.

Además de las tareas fundamentales descritas anteriormente, otra tarea esencial que suele formar parte de un típico *pipeline* de *NLP* es el reconocimiento de entidades nombradas o *Named Entity Recognition* (*NER*).

***Named Entity Recognition* (*NER*):** Esta técnica consiste en la identificación y clasificación de entidades nombradas dentro de un texto en función de su categoría. Una entidad nombrada es cualquier objeto o concepto específico del mundo real que es identificado y mencionado con un nombre propio, como personas, lugares, organizaciones, fechas, cantidades, etc. El objetivo del *NER* es detectar y etiquetar automáticamente estas entidades importantes, para comprender mejor el contenido del texto y facilitar su análisis semántico. Por ejemplo, al aplicar técnicas de *NER* en la oración "El actual presidente de Chile, Gabriel Boric, es egresado de derecho de la Universidad de Chile", se identifica "Gabriel Boric" como una persona, "Chile" como un lugar y "Universidad de Chile" como una organización.

2.2.2. Procesamiento de Texto con Redes Neuronales

El procesamiento de texto con redes neuronales se centra en el uso de modelos basados en redes neuronales para comprender, analizar y generar datos de texto. A través de estos modelos, las máquinas pueden procesar y analizar grandes cantidades de texto de manera eficiente, y en muchos casos obtener resultados de calidad cercanos a los humanos.

Una tarea fundamental para poder utilizar redes neuronales para el procesamiento de texto, es convertir las entradas de texto plano en valores que las redes neuronales puedan entender. Es así que los *tokenizadores* se encargan de transformar el texto en *tokens*. Los cuales, como se mencionó anteriormente, son divisiones de texto que se hacen de manera inteligente, y que pueden ser caracteres, sub-palabras o palabras completas de un párrafo. Luego de *tokenizar* el texto plano, los *tokens* son convertidos en vectores numéricos densos en un espacio de alta dimensión, los cuales sirven como entrada a las redes neuronales.

Sin embargo, estas representaciones en vectores no siempre son las mejores. Por ejemplo, si se usa *one-hot encoding* [27] para la representación de los vectores numéricos, se está asumiendo que todas las palabras tienen el mismo grado de similitud, y se está perdiendo la dimensión que entrega información sobre qué palabras tienen una relación conceptual entre ellas. Por lo cual es importante elegir buenas representaciones, los *words embeddings* [4] son representaciones compactas de texto representadas en vectores de alta dimensionalidad, pero que a la vez mantienen las relaciones conceptuales entre palabras. (Ver figura 2.6).

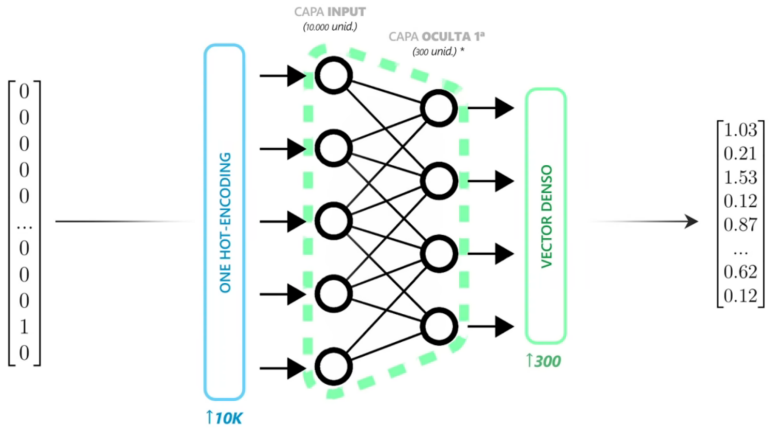

Figura 2.6: Ejemplo de *embedding* generado a partir de una entrada en *one-hot encoding* [5].

Algunos métodos populares para generar *words embeddings* son *Word2Vec*, *GloVe* y *FastText*, los cuales han demostrado ser bastante efectivos para la captura de relaciones semánticas y sintácticas entre palabras, y representarlas en vectores de alta dimensionalidad. Permitiendo que el texto plano sea convertido en representaciones de un espacio vectorial.

Una vez que se han obtenido los *word embeddings*, se abre un mundo de posibilidades para el procesamiento del lenguaje natural mediante el uso de redes neuronales. Los vectores de alta dimensionalidad pueden servir como entrada para modelos de *deep learning* en diversas tareas de *NLP* relacionada con el texto, como el reconocimiento y la clasificación de texto, la traducción automática, el resumen de texto, la generación de lenguaje natural, el análisis de sentimientos, entre otras. La arquitectura de las *DNN* utilizadas en *NLP* varía según la tarea específica, pero a menudo se utilizan:

Redes neuronales recurrentes (*RNN*): Las *RNN* son un tipo de red neuronal especialmente diseñado para trabajar con datos secuenciales, como texto, audio, series temporales, etc. La idea principal detrás de las *RNN* es que tienen conexiones recurrentes que les permiten mantener una especie de "memoria" o estado oculto a medida que procesan cada elemento en una secuencia, permitiéndole recordar información relevante de pasos de tiempo anteriores a lo largo de la secuencia. Este estado oculto es actualizado en cada paso de tiempo y se combina con la entrada actual para obtener una salida, lo que las convierte en redes especialmente adecuadas para modelar la estructura secuencial del lenguaje. Sin embargo, las *RNN* tradicionales tienen un problema conocido como "desvanecimiento del gradiente" o "explosión del gradiente". Esto ocurre cuando el gradiente (la medida de la pendiente para ajustar los pesos) se vuelve demasiado pequeño o demasiado grande a medida que se propaga hacia atrás en el tiempo durante el entrenamiento. Esto hace que las *RNN* tengan dificultades y sufran limitaciones al momento de guardar relaciones con secuencias demasiado largas. Tomando el caso del texto, a medida que se avanza en la secuencia se desvanece la relación existente con las primeras palabras, las cuales podrían estar aportando un alto contenido semántico, incluso cambiando totalmente el significado de un texto.

Redes neuronales de memoria a corto y largo plazo (*LSTM*) y redes neuronales de compuertas (*GRU*): Las *LSTM* y las *GRU* son dos arquitecturas extensoras de las *RNN*, y que fueron diseñadas para abordar el problema del desvanecimiento del gradiente.

Estas arquitecturas incluyen mecanismos de compuertas que regulan el flujo de información dentro de la red, lo que les permite recordar y olvidar información a lo largo del tiempo de una manera más efectiva, siendo especialmente útiles para capturar dependencias a largo plazo en datos secuenciales como el texto. Específicamente:

- Las *LSTM* introducen una celda de memoria que puede mantener información a lo largo de la secuencia. La celda de memoria es actualizada mediante una serie de operaciones con compuertas, como las compuertas de olvido, de entrada y de salida. Las compuertas controlan qué información debe olvidarse, qué información debe agregarse a la celda de memoria y qué información debe ser la salida.
- Las *GRU* son una variante más simple de las *LSTM* pero aún eficaz. Tienen menos compuertas y menos parámetros, lo que las hace más fáciles de entrenar en conjuntos de datos más pequeños. Las *GRU* también tienen una celda de memoria y compuertas de actualización y de reinicio que controlan cómo se actualiza el estado oculto y cómo se combina con la entrada actual.

Debido a su naturaleza recurrente y dependencia en pasos de tiempo anteriores, las *RNN*, *LSTM* y las *GRU* son más difíciles de paralelizar durante el entrenamiento, lo que puede afectar negativamente su rendimiento en tareas que se benefician de la paralelización. Además, a pesar de que las *LSTM* y las *GRU* cuentan con mecanismos que ayudan a mitigar el problema del desvanecimiento del gradiente, todavía pueden sufrir de gradientes que explotan o desaparecen en secuencias muy largas.

Redes neuronales convolucionales (*CNN*) para texto: Si bien las redes neuronales convolucionales son ampliamente conocidas por su aplicación en tareas de visión por computadora, también se han adaptado al procesamiento de texto para extraer características relevantes. En el contexto del procesamiento de texto, las *CNN* aplican filtros convolucionales sobre ventanas de palabras en una oración o secuencia de texto. A medida que el filtro se desplaza a lo largo de la secuencia, se extraen características locales y patrones relevantes del texto en cada posición, siendo eficaces por ejemplo para la detección de *n-gramas* y características gramaticales. Estas características locales se combinan y reducen a través de capas de agrupación (*pooling*) para obtener una representación global del texto que se alimenta a través de una o varias capas densas para realizar la tarea deseada. Sin embargo, estas redes pueden perder la información secuencial crucial para comprender el significado y la coherencia del texto, ya que los filtros convolucionales no pueden capturar relaciones de dependencia a largo plazo entre palabras. Además, si se utilizan filtros convolucionales de gran tamaño o varias capas en una *CNN* para texto, el número de parámetros puede aumentar considerablemente. Esto puede resultar en un alto consumo de memoria y poder de cómputo, dificultando su entrenamiento y uso en dispositivos con recursos limitados.

Resumiendo, las *RNN* son efectivas para trabajar con secuencias de texto, pero pueden sufrir de desvanecimiento del gradiente. Las *LSTM* y las *GRU* abordan este problema al incluir mecanismos de compuertas para controlar el flujo de información, pero tienen problemas con la paralelización en su entrenamiento. Mientras que las *CNN* para texto son capaces de extraer características locales, pero no pueden capturar relaciones de dependencia a largo plazo entre palabras y sufren de un crecimiento gigantesco de parámetros, lo que dificulta su entrenamiento.

Otros enfoques son utilizados para comprender frases o documentos más largos. En estos casos se utilizan técnicas como el modelo de bolsa de palabras (*Bag-of-Words*), que representa un documento como un conjunto de palabras sin considerar su orden, centrándose únicamente en la frecuencia de ocurrencia de las palabras.

Sin embargo, no hay lugar a duda de que, en cuanto al estado del arte se refiere, las técnicas más recientes y con mejor éxito en procesamiento del lenguaje natural son las que guardan relación con los mecanismos de atención. A continuación, se procederá a explicar en qué consisten estas técnicas y cómo han revolucionado el campo del *NLP*.

2.2.3. *Transformers*

Los mecanismos de atención llegan a solucionar los problemas anteriormente presentados. Estos mecanismos con capaces de "prestar mayor atención" a diferentes aspectos de la secuencia de datos de entrada durante el proceso de codificación y decodificación, lo que es utilizado para identificar a que componentes darles mayor relevancia en una secuencia, y así capturar relaciones más complejas y aprender patrones de forma más eficiente. Esto se logra mediante el uso de un conjunto de mecanismos de aprendizaje que permiten que la red se centre en algunas partes de los datos y olvide otras.

La arquitectura por excelencia que implementa mecanismos de atención y que ha revolucionado el campo del *NLP* desde su introducción es el *Transformer* [32] (Ver Figura 2.7).

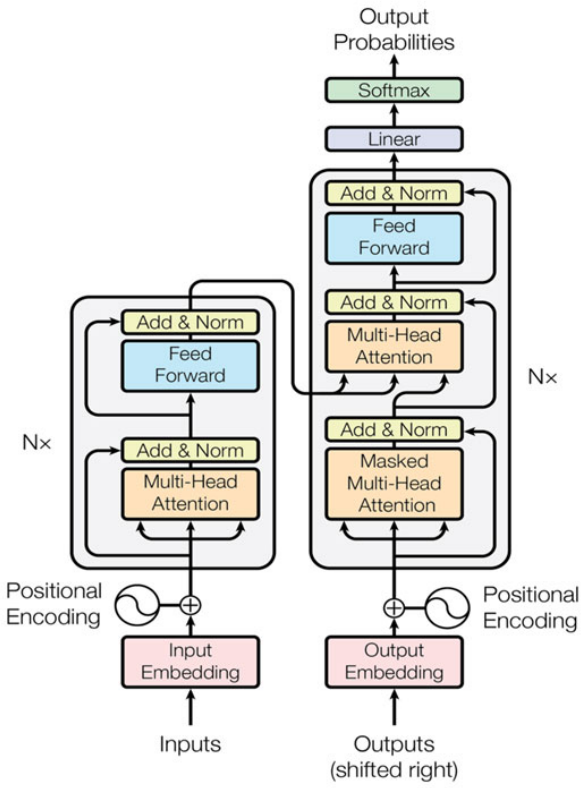

Figura 2.7: Modelo de arquitectura de un *Transformer* [32].

Los *Transformers* son una arquitectura de redes neuronales para secuencias basadas en mecanismos de autoatención, que han demostrado funcionar bien para el procesamiento de texto y que se encuentran actualmente impulsando importantes avances en el campo del *NLP*. Estos modelos están diseñados para aprender representaciones semánticas de palabras, frases y oraciones mediante el uso de *DNN*. A diferencia de las arquitecturas recurrentes anteriores, los *Transformers* han demostrado ser altamente efectivos al capturar relaciones o dependencias de largo alcance en las secuencias y capturar la semántica global de los textos.

Gracias a los mecanismos de atención, la red puede enfocarse en partes relevantes del texto durante el proceso de codificación y decodificación. Al asignar diferentes ponderaciones a las palabras en función de su relevancia contextual, los *Transformers* pueden detectar las conexiones más significativas en el discurso y lograr una comprensión más profunda en su significado, lo que les permite capturar relaciones de largo alcance en el texto sin depender de una secuencia ordenada de entrada. En lugar de procesar la secuencia de palabras de manera lineal, como en las redes neuronales recurrentes, los mecanismos de atención pueden analizar todas las palabras en paralelo, otorgando mayor importancia a ciertas palabras en función de la relevancia para la tarea en cuestión. Esto significa que pueden procesar palabras o elementos de manera paralela, lo que conduce a una mayor eficiencia en el entrenamiento.

Esta arquitectura ha sido especialmente exitosa en tareas como traducción automática, generación de texto, resumen de documentos, reconocimiento de entidades, clasificación de texto, entre otras. Su capacidad para modelar dependencias a largo plazo y capturar la semántica global del texto ha mejorado significativamente la precisión y la calidad de las predicciones en diversas aplicaciones del *NLP*, pero también ha demostrado su eficacia en otras áreas de aprendizaje profundo, como el procesamiento de imágenes y la generación de música, lo que muestra su versatilidad y capacidad para adaptarse a diversas tareas.

Los *Transformers* han revolucionado el campo del *NLP* y han demostrado ser una arquitectura poderosa y eficaz para abordar una amplia gama de problemas en el aprendizaje profundo. Su capacidad para modelar dependencias a largo plazo y aprovechar la atención ha abierto nuevas posibilidades en la generación y comprensión de lenguaje, y ha desencadenado avances significativos en la inteligencia artificial en general.

Esta arquitectura se compone de dos partes fundamentales: El *encoder* y el *decoder*.

- ***Encoder*:** Se encarga de recibir la secuencia de entrada y de generar una representación numérica contextualizada o *embeddings* contextuales para cada *token*, capturando la semántica de una frase en el proceso.
- ***Decoder*:** Recibe los *embeddings* del codificador y produce secuencias de salida para una tarea específica. Se utiliza en tareas de generación de lenguaje, como traducción automática o resumen de texto.

Un ejemplo práctico de esto podría ser una tarea de traducción automática entre los idiomas español e inglés. En ella, el *encoder* se encargaría de aprender la semántica del idioma español, a través de la generación de *embeddings* contextuales de las palabras. Mientras que el *decoder* se encargaría de aprender la correlación entre las palabras del español y el inglés, generando las salidas de la traducción [1].

2.2.4. Modelos del Lenguaje

Una de las técnicas más populares en *NLP* es el uso de redes neuronales preentrenadas, como los modelos de lenguaje basados en *Transformers*. Los modelos de lenguaje [28] son modelos multi-tarea no supervisados [12] entrenados con grandes volúmenes de datos de texto, siendo capaces de realizar variadas tareas con buenos resultados. Algunos ejemplos de modelos de lenguaje son *BERT* (*Bidirectional Encoder Representations from Transformers*) y *GPT* (*Generative Pre-trained Transformer*). Estos modelos son entrenados con grandes cantidades de texto sin etiquetar y pueden capturar información contextual y semántica profunda.

Por un lado, *BERT* fue preentrenado con texto en inglés extraído de fuentes como *Wikipedia* y *BooksCorpus* [8]. Su principal ventaja es que se utiliza un proceso de pre-entrenamiento bidireccional, lo que le permite capturar el contexto de las palabras tanto hacia adelante como hacia atrás en una oración. Esto ayuda a que el modelo comprenda mejor la relación entre las palabras y el contexto global y es especialmente útil en tareas de comprensión del lenguaje, donde el significado de una palabra puede depender del contexto en el que se encuentra.

Por otro lado, *GPT* es un modelo generativo, lo que significa que puede generar texto de manera coherente y cohesiva. Esto ha llevado a su uso en aplicaciones de generación de texto como *Chatbots*. De hecho, *ChatGPT*, que es una implementación específica de *GPT* desarrollada por *OpenAI*, ha sido ampliamente reconocida y utilizada como una herramienta avanzada para crear *chatbots* conversacionales.

A través del entrenamiento en grandes conjuntos de texto y distintos idiomas, *ChatGPT* ha adquirido un conocimiento amplio y diverso del lenguaje, lo que le permite responder a una variedad de preguntas y mantener conversaciones significativas con los usuarios.

Es importante destacar que *ChatGPT*, al igual que otros modelos, son ejemplos de cómo la generación de texto coherente y creativa puede ser aplicada para mejorar la interacción humano-máquina. Sin embargo, debido a sus capacidades para generar contenido de manera autónoma, también es importante tener en cuenta la necesidad de controlar y supervisar el uso de estas tecnologías para evitar la generación de contenido inapropiado o engañoso.

También existen modelos del lenguaje que se inspiran en otros, utilizando en sus arquitecturas combinaciones de componentes similares a las de otros modelos. Un ejemplo de esto es *BART* (*Bidirectional and Auto-Regressive Transformers*) [16], este modelo del lenguaje desarrollado por *Facebook AI Research* (*FAIR*) combina elementos tanto de *BERT* como de *GPT*, pero sus componentes y módulos en específico son un poco diferentes.

La arquitectura de *BART* se basa en un enfoque *encoder-decoder*, donde el *encoder* utiliza un conjunto de capas de atención de *transformers* bidireccionales para procesar la entrada de manera que pueda capturar información contextual de ambas direcciones (hacia adelante y hacia atrás), y el *decoder* utiliza un conjunto de capas de atención de *transformers* autoregresivos, donde cada palabra se predice secuencialmente basándose en las palabras generadas previamente en la secuencia de salida. Por lo cual, *BART* combina un *encoder* bidireccional similar a *BERT* con un *decoder* autoregresivo similar a *GPT*. Esta combinación permite que *BART* sea eficiente en la generación de texto de alta calidad y lo hace especialmente

adecuado para tareas de comprensión, generación y traducción de lenguaje natural.

Otros modelos con buenos desempeños en tareas de traducción automática o *machine translation* (*MT*) son los modelos desarrollados por el proyecto *OPUS-MT*[1]. Este proyecto se enfoca en el desarrollo de *software* libre y herramientas para la traducción automática. Su misión es proporcionar servicios de traducción que estén libres de intereses y restricciones comerciales, haciendo accesible la traducción automática para cualquier persona de forma transparente y segura, sin planes de explotación ni comprometiendo su privacidad.

Los modelos entrenados por este grupo se basan en la traducción automática neuronal o *neural machine translation* (*NMT*) haciendo uso de arquitecturas *transformers* de última generación. Para esto, en su marco de trabajo utilizan *Marian-NMT2*[2], un *toolkit* para *NMT* estable, con capacidades eficientes de entrenamiento y decodificación, y preparado para poner en producción los modelos desarrollados [30].

El estado actual del proyecto se encuentra en un repositorio[3] con más de 1000 modelos de traducción neuronal preentrenados, listos para ser lanzados en servicios de traducción en línea. Para conseguir esto, también proporcionan implementaciones *open source* de aplicaciones *web* que pueden ser ejecutadas de manera eficiente en *hardware* de escritorio promedio, con una configuración e instalación sencillas.

Una de las características que comparten en común todos los modelos mencionados, es que al ser preentrenados en grandes cantidades de texto sin etiquetar, y al ser capaces de capturar información contextual y semántica profunda, se logra que los modelos "entiendan el lenguaje" y que tengan un buen punto de partida para aprender a realizar otras tareas.

En otras palabras, estos modelos pueden ser especializados en tareas específicas, aplicando técnicas de ajuste fino o *fine-tuning* [14], para mejorar desempeños en tareas comunes de *NLP* como el análisis de dependencias, el reconocimiento de entidades, el análisis de sentimientos, la generación de texto, traducción automática, entre otras.

El *fine-tuning* consiste en que el modelo se entrena en un conjunto de datos más pequeño y etiquetado para adaptarse a la tarea deseada. Por lo cual, la capacidad de los modelos para comprender el lenguaje es muy importante, ya que determinará que tanto más se puede entrenar el modelo para un propósito específico.

La revolución de los modelos de lenguaje, y el concepto de *fine-tuning*, permite que los desarrolladores puedan construir aplicaciones sobre modelos preentrenados por otros, en vez de comenzar desde cero cada vez. Además, el paralelismo permite tiempos de entrenamiento más cortos, resultando en modelos de mejor calidad y precisión.

Hugging Face es una compañía que permite usar los últimos y mejores modelos relacionados con el *NLP* en forma gratuita, lo que ha provocado un efecto positivo en los años recientes, a nivel de usuario y empresas. Dado que su librería es *open source*, presenta al mundo una oportunidad de desarrollo abierto y colaborativo.

[1] https://github.com/Helsinki-NLP/Opus-MT
[2] https://marian-nmt.github.io
[3] https://github.com/Helsinki-NLP/

En el contexto de este trabajo se pretende usar los mejores modelos de lenguaje relacionados con el *NLP* para analizar y obtener métricas sobre los relatos ingresado por los usuarios de la herramienta. Si bien estos modelos no son perfectos, la idea clave es planificar una arquitectura y un sistema que permita los objetivos planteados, pero que a la vez sea lo suficientemente flexible para cambiar o añadir modelos a medida que estos mejoran con el tiempo.

2.3. Modelos Generativos

Los modelos de inteligencia artificial generativos son modelos que se utilizan para generar nuevos datos, como imágenes, texto, música o incluso videos, que se asemejan a los datos de entrenamiento proporcionados previamente. Estos modelos son capaces de aprender patrones y estructuras subyacentes en los datos de entrenamiento y luego utilizar ese conocimiento para generar nuevas muestras que son similares en estilo, contenido y distribución a los datos originales.

Existen varios tipos de modelos generativos en inteligencia artificial, pero los enfoques más populares y exitosos son:

- **Redes Generativas Adversarias (*GAN*):** En un modelo *GAN* hay dos componentes principales, el generador y el discriminador. El generador toma una distribución de ruido aleatorio como entrada e intenta generar muestras sintéticas similares a la data de entrenamiento. El discriminador, por otro lado, evalúa las muestras generadas por el generador y las compara con las muestras reales del conjunto de entrenamiento. El objetivo es que el generador mejore continuamente su capacidad para generar muestras hasta que el discriminador no pueda distinguirlas de las reales. Uno de los problemas asociados a estos modelos es el sobre entrenamiento, cuando sucede esto el generador produce salidas con poca diversidad y originalidad.
- **Modelos Variacionales *Autoencoders* (*VAE*):** En un modelo *VAE*, se utiliza una arquitectura conformada por un *encoder* y un *decoder*, con el objetivo de aprender una representación latente del conjunto de datos de entrada, y a partir de esta representación generar nuevos datos. El *encoder* mapea los datos de entrada de alta dimensionalidad en una representación de baja dimensión, mientras que el *decoder* intenta reconstruir los datos de entrada originales de alta dimensión mapeando esta representación de vuelta a su forma original. Uno de los problemas asociados a este tipo de modelos es que dado que el *encoder* aprende una representación latente de los datos, ocurre el caso en que dos distribuciones de representación latentes se solapan entre sí, provocando que la salida óptima del *decoder* sea el promedio de las dos entradas, lo que conduce a resultados con muestras borrosas.

Estos modelos generativos tienen aplicaciones en diversas áreas, como la generación de imágenes y arte generativo, la creación de música y composición automática, la síntesis de voz y la generación de texto. También se utilizan para mejorar tareas de reconocimiento y

clasificación, como el aumento de datos de entrenamiento (*data augmentation*) o la generación de ejemplos adversarios para probar la robustez de otros modelos de inteligencia artificial.

2.3.1. Modelos de Difusión

Un tercer tipo de modelo generativo con las mismas aplicaciones y que rivaliza con los modelos mencionados anteriormente son los modelos de difusión. Estos modelos son redes neuronales profundas que contienen variables latentes capaces de aprender la estructura de una imagen determinada, eliminando el ruido generado en ella.

En estos modelos hay dos componentes principales, un proceso *forward diffusion* y uno de *reverse diffusion*. El primer proceso consta de varios pasos en los cuales se agrega gradualmente una pequeña cantidad de ruido gaussiano a la muestra hasta que se convierte en ruido blanco. Por otro lado, el proceso de *reverse diffusion* tiene como objetivo revertir el proceso de *forward diffusion*, paso a paso eliminando el ruido para recuperar los datos originales.

Este concepto es similar a un modelo *VAE* en la forma en que trata de optimizar una función objetivo, primero proyectando los datos en el espacio latente y luego recuperándolos al estado inicial. Sin embargo, en lugar de aprender la distribución de datos, el sistema tiene como objetivo modelar una serie de distribuciones de ruido en una cadena de *Markov* [23] y decodificar los datos deshaciendo o eliminando el ruido de forma jerárquica.

A diferencia de un *VAE* y una *GAN*, los cuales generan muestras en un solo paso, los modelos de difusión crean muestras paso a paso. El modelo primero crea una estructura base de imagen y luego se enfoca en agregar detalles finos en la parte superior iterativamente.

A continuación podemos visualizar una comparativa entre los comportamientos y funcionamientos de los distintos modelos generativos que fueron descritos (Ver Figura 2.8).

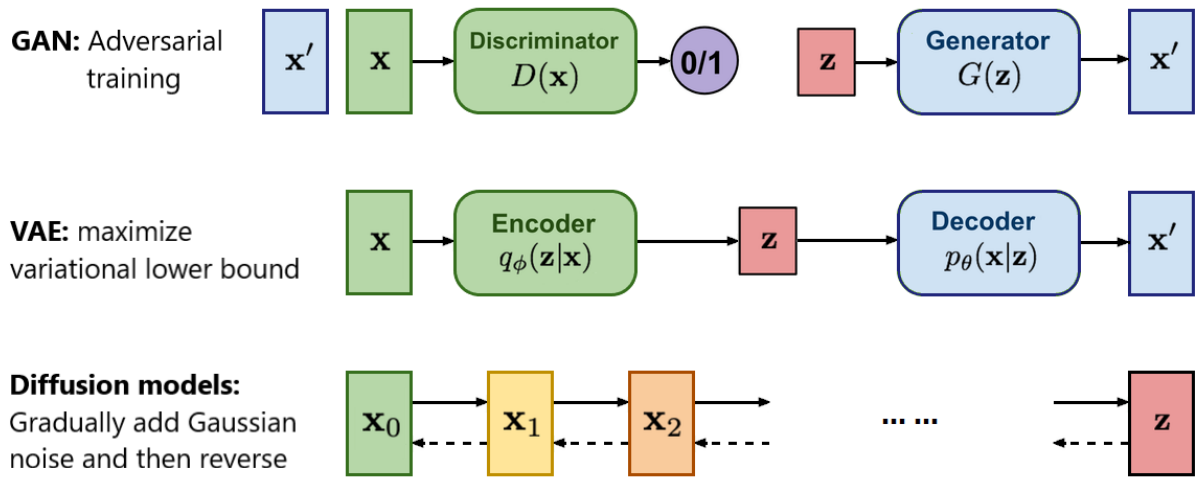

Figura 2.8: Comparativa entre Modelos Generativos [11].

Una vez que un modelo de difusión está entrenado para entender la abstracción del con-

cepto detrás de una imagen, puede crear nuevas variaciones de esa imagen. Por ejemplo, al eliminar el ruido de una imagen de un gato, el modelo de difusión visualiza una imagen limpia del gato, aprende cómo se ve el gato y aplica este conocimiento para crear nuevas variaciones de la imagen del gato (Ver Figura 2.9).

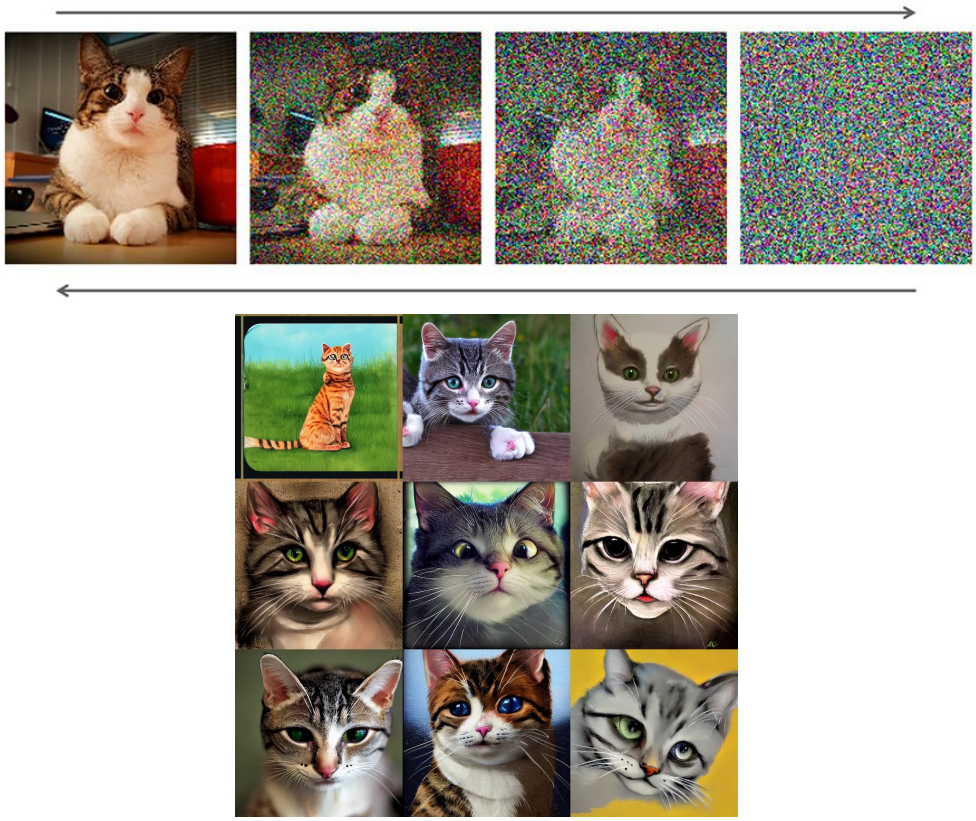

Figura 2.9: Representación del comportamiento de un modelo de difusión [19].

Algunos sistemas de generación automática de imágenes pioneros en la utilización de modelos generativos son *DALL-E 2* [4], *Midjourney* [5] y *Stable Diffusion* [6].

Este último es de código abierto y puede ser ejecutado de múltiples formas, ya que gracias a la comunidad *open source* constantemente aparecen a la luz variantes del modelo y formas de ejecución. Algunas variantes hacen uso de *notebooks* de *Google Colab*, otras integran interfaces gráficas, e incluso algunas aplican optimizaciones para requerir de menos poder computacional, lo que permite que el modelo pueda ser utilizado por más gente y de diversas formas, no restringiéndose solo al público que goza de *GPUs* de gama alta.

La capacidad de estos sistemas para generar imágenes de alta calidad en un par segundos a partir de descripciones de texto, conocidas como *prompts*, ha generado un enorme interés en el público y ha causado furor en las redes sociales. Esto se debe a las múltiples posibilidades y aplicaciones que ofrecen estas imágenes generadas por los modelos, revolucionando la industria creativa y de entretenimiento como nunca antes, abriendo un abanico de oportunidades

[4]https://openai.com/dall-e-2
[5]https://docs.midjourney.com
[6]https://stability.ai/stablediffusion

en el mundo creativo, permitiendo desarrollar arte visual único, lo que promete enriquecer aún más nuestras experiencias en el entretenimiento y la cultura en general.

Siguiendo los alineamientos del proyecto, se utilizarán estos modelos para apoyar ilustrativamente los relatos ingresados por los usuarios, aportándoles un toque visual único y personalizado. La capacidad de generar imágenes de alta calidad a partir de descripciones de texto permitirá que cada relato cobre vida con ilustraciones hechas a medida, capturando escenas, entidades y ambientes de manera sorprendente, y brindándole al usuario distintas perspectivas visuales de como es percibido su relato.

2.4. Trabajos Relacionados

En el contexto de proyectos, aplicaciones o estudios con objetivos similares a los de *Textarium*, es posible encontrar la existencia de algunos trabajos que hacen uso de herramientas de *NLP* para el análisis y la extracción de propiedades que caractericen descripciones de texto o relatos generados por personas.

Dentro de algunos de estos trabajos encontramos el estudio realizado por Alessandro Fogli, Luca Maria Aiello y Daniele Quercia [10], en el cual se diseñó una herramienta que es capaz de caracterizar automáticamente informes de sueños de las personas. O el proyecto en el cual se diseñó y desarrolló "*The Dreamcatcher*" [3] (Ver figura 2.10), una herramienta visual interactiva que explora el vínculo entre los sueños y la vida diaria, permitiendo la exploración de patrones de sueño y sumergirse en recopilaciones individuales de personas que registraron sus sueños, aumentando la conciencia del público en general sobre los beneficios del análisis onírico.

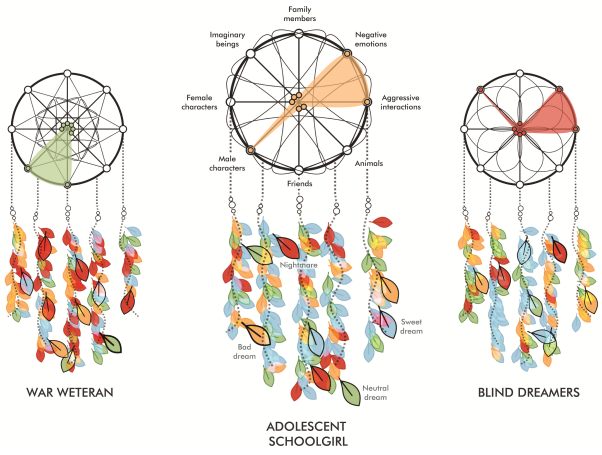

Figura 2.10: The Dreamcatcher: una herramienta visual interactiva que explora el vínculo entre los sueños y la vida de las personas [3].

Por el lado de herramientas que brinden apoyo directo a la escritura, se pueden mencionar proyectos como *SudoWrite*[7], una plataforma de redacción asistida por inteligencia

[7]https://www.sudowrite.com

artificial que ayuda a los usuarios a generar contenido de manera más rápida y efectiva utilizando tecnologías y técnicas de procesamiento de lenguaje natural avanzadas. Una de las caracteristicas de *SudoWrite* es que también puede ayudar a los usuarios a superar bloqueos creativos al proporcionar sugerencias y recomendaciones para la redacción de textos. Esto puede ser especialmente valioso para escritores y creadores que buscan inspiración o nuevas ideas.

Sin embargo, *SudoWrite* se limita a esto y no explora más alternativas, no integra modelos relacionados con la visualización e ilustración, ni realiza análisis profundos y descomposiciones sintácticas y/o semánticas de texto. Por lo cual *Textarium*, al tener como propósito la integración de un conjunto de funcionalidades base que toman en consideración estos aspectos, se puede destacar integrando de buena manera tanto análisis directos sobre el texto, como análisis más extractivos, analíticos, y por supuesto, visuales.

Cabe mencionar que, para poder acceder y utilizar las funcionalidades de *SudoWrite*, al igual que con muchas otras aplicaciones similares, se deben realizar pagos de dinero por el servicio. Por lo cual, proyectos abiertos a la comunidad *open source*, como es el caso de *Textarium*, se destacan debido a su ventaja competitiva, siendo apoyados ampliamente por la comunidad, logrando competir de igual a igual con aplicaciones de pago.

Por otro lado, también existen proyectos centrados en la utilización de modelos de difusión para la ilustración y enriquecimiento de textos. Uno de estos casos es el proyecto llamado *Long Stable Diffusion: Long-form text to images*[8], el cual utiliza *GPT-3* para generar resúmenes de los relatos o cuentos procesados y crear *prompts* con estilos predefinidos a partir de estos, las cuales terminan transformándose en una serie de ilustraciones secuenciales del texto ingresado gracias al modelo de difusión.

El tema con este proyecto es que al utilizar *GPT-3* como modelo de lenguaje, se debe utilizar la *API* de *OpenAI* para realizar las peticiones al servicio de *GPT-3*. Dicho esto, si bien esta *API* brinda una cierta cantidad de *tokens* gratis para que pueda ser utilizada libremente, al momento de consumirse el saldo cada petición comienza a ser de pago.

Con todo esto en mente, si bien existen proyectos con foco en la misma temática, y que reúnen características y funcionalidades bastante similares a las de *Textarium*, este es capaz de diferenciarse y destacarse de los demás debido a su cualidad de proyecto *open source* y de *MVP*, debido a que esto promueve su accesibilidad y el hecho de que pueda ser adoptado y modificado por una comunidad amplia de desarrolladores y usuarios, lo que potencia enormemente su utilidad y flexibilidad.

[8]https://github.com/sharonzhou/long_stable_diffusion

Capítulo 3

Diseño de la Solución

En este capítulo se describirá el proceso de diseño y planeación de la arquitectura de la herramienta de análisis y visualización de textos.

El objetivo principal es proporcionar al lector una comprensión integral de lo que se hizo en el desarrollo de la aplicación. Para lograr esto, se presentará la aplicación y la estructura lógica detrás de ella, pero sin especificar detalles de la implementación, como herramientas de desarrollo, modelos y algoritmos utilizados, etc. Estos detalles, y más cuestiones relacionadas con la implementación, serán descritas en el siguiente capítulo, el cual está estrictamente relacionado con la implementación del sistema.

La estructura de este capítulo está hecha de una manera similar a la estructura de la aplicación, de manera que sea más fácil de entender para el lector. La sección 3.1 describe los requisitos de *software* de la aplicación, la sección 3.2 presenta la arquitectura general de la aplicación, y las siguientes subsecciones presentan cada uno de los módulos que conforman la herramienta: La sección 3.2.1 describe el Módulo de Preprocesamiento, la sección 3.2.2 explica el Módulo de Extracción de Conceptos Claves, la sección 3.2.3 explica el Módulo de Detección de Entidades, la sección 3.2.4 describe el Módulo de Visualización Semántica del Texto, y finalmente la sección 3.3 presenta el diseño de la interfaz que logra integrar estos módulos para que se comporten como una aplicación usable por usuarios.

3.1. Requisitos de Software

Antes de iniciar con el desarrollo de cualquier componente de una aplicación de *software* es necesario tener clara la definición e implicancias del problema que se busca resolver.

En este caso, la problemática que se busca resolver está relacionada con los bloqueos creativos que los escritores sufren, y las consecuencias que estos traen, como la falta de inspiración e ideas atractivas, dificultades para encontrar palabras adecuadas y evocadoras que les permite expresarse de manera precisa y efectiva, entre otras.

Para solucionar esto, la idea es proporcionar a los escritores una herramienta innovadora, accesible y fácil de usar que les permita potenciar su creatividad y apoyarlos en su escritura de manera más eficiente, ayudándolos a superar estos obstáculos y perfeccionar su proceso creativo. Específicamente, esta aplicación busca facilitar la identificación de palabras clave, la comprensión de las entidades y las relaciones entre los elementos del texto, además de brindar apoyo ilustrativo al relato que se esté analizando.

Por lo tanto, el sistema debe cumplir con los siguientes requisitos:

1. **Análisis y comprensión de las entradas de texto:** El sistema debe ser capaz de analizar y comprender el texto proporcionado por los usuarios. Esto implica comprender la estructura y contenido del texto, así como también la capacidad de identificar y extraer palabras frecuentes, conceptos clave e ideas relevantes.
2. **Generación de contenido relevante:** Para que la herramienta agregue valor real, debe generar contenido adicional que sea relevante y útil para los usuarios. Esto incluye el cómo es percibido y entendido el contenido del texto, la detección de entidades y sugerencias de entidades relacionadas, e ilustraciones que puedan enriquecer el texto y potenciar la creatividad del escritor. Al ofrecer estas opciones creativas, la herramienta busca estimular la imaginación de los usuarios para que estos puedan encontrar nuevas formas de expresión.
3. **Fácil utilización y presentación del contenido:** La herramienta debe ser fácil de utilizar, y debe presentar el contenido generado y sugerencias de manera clara y organizada. Esto implica mostrar los conceptos claves, entidades identificadas, imágenes que apoyen la comprensión e ilustración del relato, y en general cualquier contenido relevante detectado o generado en una interfaz fácil de usar y accesible. Así, los usuarios pueden tener una visión clara de los elementos esenciales de sus relatos y comprender mejor la estructura y percepción de su trabajo.

Estos requisitos garantizan que la herramienta proporcione un análisis profundo, opciones creativas y una presentación clara del contenido generado, ayudando a los escritores a superar bloqueos creativos y potenciar su proceso de escritura de manera efectiva

Dicho esto, se procederá a explicar la arquitectura general de la aplicación, para luego entrar en detalle con los módulos que abordan los requerimientos necesarios para cumplir los requisitos listados.

3.2. Arquitectura General

Debido a la naturaleza y orientación de la aplicación, no es necesario realizar una búsqueda de un *dataset*, ni una recopilación de textos con los cuales trabajar. Los textos que se entregarán como *input* y que serán procesados son brindados directamente por los usuarios finales. Por lo cual la principal preocupación y foco de la herramienta está en el análisis y comprensión de estas entradas.

Para asegurar este requisito, se debe tener un procesamiento sólido que sea capaz de realizar las siguientes tareas:

1. **Análisis lingüístico y gramatical:** La herramienta debe contar con algoritmos y técnicas avanzadas para realizar un análisis lingüístico exhaustivo de los textos proporcionados por los usuarios finales. Esto implica identificar la estructura gramatical, como las partes del discurso (sustantivos, verbos, adjetivos, etc.), las relaciones sintácticas y las reglas gramaticales aplicadas.
2. **Extracción de información relevante:** El procesamiento debe ser capaz de extraer la información relevante contenida en los textos de los usuarios. Esto incluye identificar entidades mencionadas, como nombres de personas, lugares, organizaciones, fechas, entre otros. Además, debe ser capaz de reconocer conceptos clave y su contexto, para comprender el significado y la intención del texto.
3. **Generación de representaciones semánticas:** Es esencial que la herramienta pueda generar representaciones semánticas precisas de los textos procesados. Esto implica capturar la semántica y el significado implícito, así como las relaciones entre las palabras y las ideas expresadas en el texto.
4. **Procesamiento eficiente y escalable:** Dado que la herramienta procesará textos proporcionados por múltiples usuarios finales, es necesario que el pipeline de procesamiento sea eficiente y escalable. Debe ser capaz de manejar grandes volúmenes de datos de manera rápida y precisa, sin comprometer la calidad del análisis y la comprensión de los textos.

Construir un *pipeline* de procesamiento robusto con estas capacidades permitirá que la herramienta analice y comprenda de manera efectiva los textos proporcionados por los usuarios finales, siendo capaz de extraer información relevante, generar representaciones semánticas y brindar un procesamiento eficiente y escalable.

Dicho esto, la arquitectura del *pipeline* de procesamiento está compuesta de la siguiente manera:

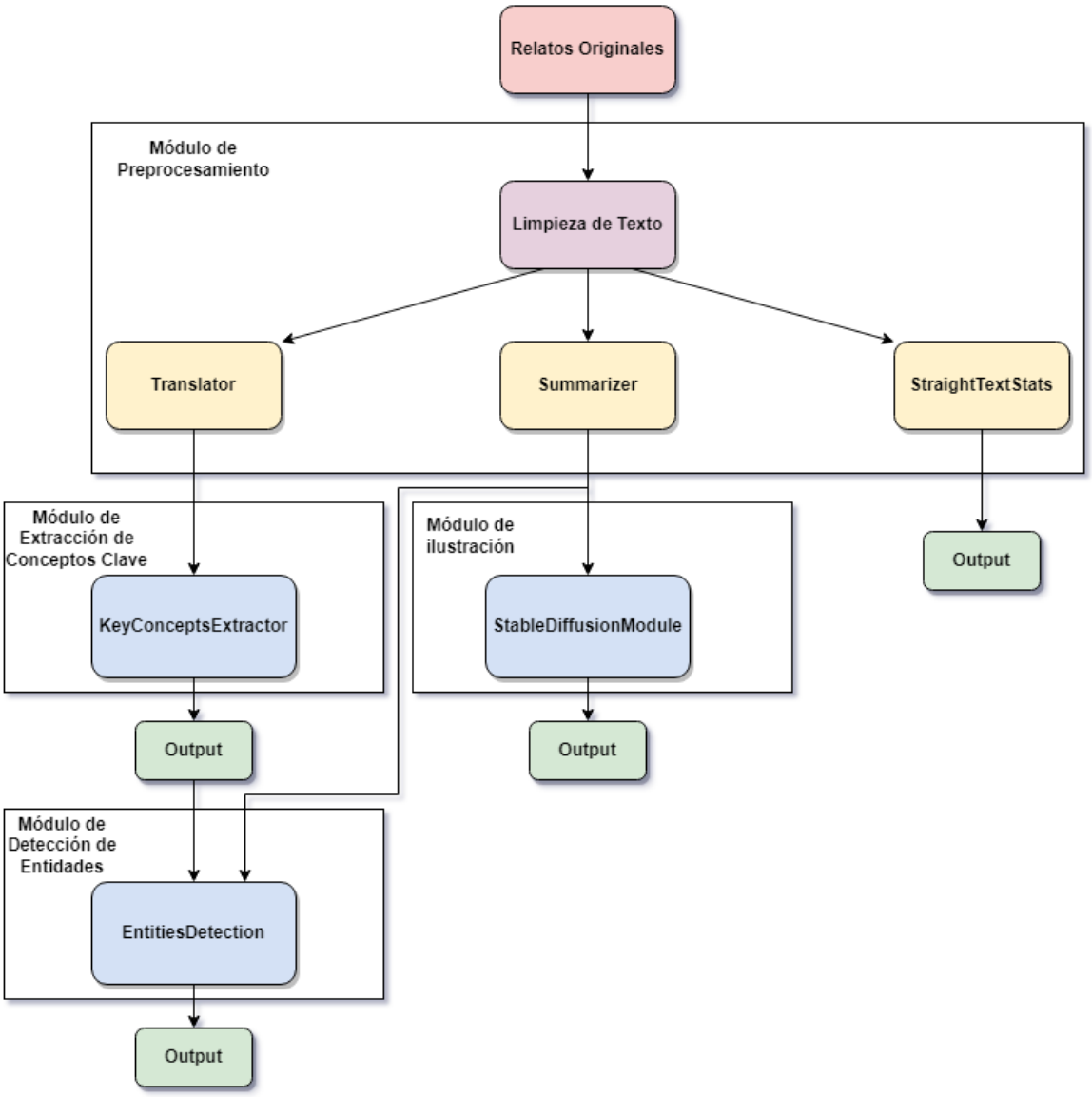

Figura 3.1: Arquitectura del *pipeline* de procesamiento.

En la figura, el rectángulo rojo corresponde a las entradas de texto ingresadas por los usuarios. El rectángulo morado corresponde a una función auxiliar de limpieza de texto, la cual dependiendo de los parámetros que recibe realiza más o menos procesamiento. Los rectángulos azules corresponden a módulos y los amarillos a submódulos, ambos son componentes de más alto nivel cuyo contenido será profundizado más adelante en la sección. Por último, los rectángulos verdes son los *outputs* del *pipeline* de procesamiento, los cuales se convertirán en los resultados finales que podrá visualizar el usuario al momento de que se completen los análisis de su relato.

A continuación se realizará una descripción de los módulos y submódulos que componen al *pipeline* de procesamiento.

3.2.1. Módulo de Preprocesamiento

El módulo de preprocesamiento es el primer módulo que conforma al *pipeline* de procesamiento, interactuando directamente con el usuario. Se encarga de recibir, preprocesar y limpiar la entrada de texto brindada por los usuarios, con el fin de estandarizarla y dejarla lista para que los demás módulos se encarguen de realizar sus tareas. Además, este módulo aprovecha de extraer un par de características y estadísticas sencillas de ejecutar luego de un procesamiento simple.

Este módulo a su vez es posible descomponerlo en 3 submódulos, los cuales serán descritos a continuación.

Translator

Este submódulo auxiliar se encarga de traducir las entradas de texto desde un idioma fuente a un idioma de destino, por lo que sirve como nexo y transformador de *inputs* para módulos o modelos que se desempeñan mejor en otros idiomas. Como se mencionó en el marco teórico, hoy en día, a pesar de los avances en el mundo de la inteligencia artificial, aún existen algunas barreras que limitan el rendimiento y desempeño de algunos modelos. Una de estas barreras es el idioma, ya que algunos modelos tienen rendimientos superiores en su idioma de creación, mientras que otros fueron entrenados para soportar el multilingüismo.

Summarizer

Este otro submódulo auxiliar tiene como principal tarea realizar resúmenes de las entradas de texto. Debido a que la herramienta está diseñada para recibir largas cantidades de texto, se debe estar preparado para mantener la comprensión de extensos relatos. Esto se logra mediante la descomposición del texto original en diversos extractos de texto, los cuales serán analizados por separado, además de resumirlos en extractos más pequeños aún, pero que mantengan las ideas fundamentales del texto original.

Este submódulo también sirve como nexo transformador de *inputs* para otros módulos del *pipeline* de procesamiento. En este caso, brinda apoyo al módulo de ilustración, resumiendo las entradas de textos en extractos más fáciles de procesar para los modelos de difusión.

StraightTextStats

Finalmente, este tercer y último submódulo auxiliar, tiene como principal tarea extraer estadísticas y métricas de fácil ejecución sobre el texto apenas procesado. Estas estadísticas son relevantes de extraer, ya que presentadas de buena forma pueden llegar a ser útiles para conseguir el objetivo de la herramienta y apoyar a los usuarios por el lado de la percepción y la comprensión del texto que se está analizando.

En este caso, el valor agregado de la herramienta viene dado por la generación de una

nube de palabras, la cual brinda una visión general y representación visual de los términos más relevantes o recurrentes en un texto. Esto puede ayudar a identificar patrones, temas o conceptos clave presentes en el relato. Además, es una forma atractiva y concisa de representar datos, ya que es capaz de resumir visualmente un tema o concepto, lo que facilita la comprensión rápida del texto. También, y a modo de complemento de lo anterior, se genera una lista con las palabras más frecuentes del texto.

3.2.2. Módulo de Extracción de Conceptos Claves

El módulo de extracción de conceptos claves tiene como principal propósito realizar precisamente lo que describe su nombre. Recibe entradas de textos que ya atravesaron el módulo de preprocesamiento, además de también haber pasado por el submódulo *Translator*, haciendo que la extracción de conceptos claves pueda realizarse con mayor facilidad.

3.2.3. Módulo de Detección de Entidades

Por su lado, el módulo de detección de entidades recibe entradas procesadas desde dos componentes. Al igual que el módulo anterior, recibe entradas de textos que ya atravesaron el módulo de preprocesamiento, pero en este caso pasan por el submódulo *Summarizer*. Además, se nutre del *output* generado por el módulo de extracción de conceptos claves. Esto es así debido a que existe una especie de consolidación y confirmación de las entidades detectadas mediante el uso de los conceptos claves previamente detectados.

3.2.4. Módulo de Ilustración

Por último, el módulo de ilustración se encarga de enriquecer el texto brindado por los usuarios con elementos visuales. Recibe entradas de textos desde el submódulo *Summarizer*, y es capaz de generar ilustraciones que representen lo que se describe en los relatos, complementando el contenido textual y brindando una experiencia más completa y atractiva para los usuarios.

3.3. Integración e Interfaz

Todo el análisis, procesamiento y resultados obtenidos pierden el sentido si es que no son integrados en un sistema coherente y accesible que pueda ser utilizado por los usuarios finales. Es por esto que, con el objetivo de facilitar el uso de la herramienta, se desarrolló una interfaz intuitiva y fácil de usar que logre integrar el *pipeline* de procesamiento. Esta integración implica combinar las salidas de los diferentes módulos para presentar al usuario un resultado final en un formato fácil de entender. Además, implica la creación de una interfaz de usuario intuitiva y amigable que permita interactuar con el sistema, proporcionando opciones de entrada y visualización de resultados.

Capítulo 4

Implementación de la Solución

En este capítulo se describirá en detalle el desarrollo e implementación de la aplicación propuesta como solución para brindar apoyo al proceso de redacción de los escritores. Se ahondará en los distintos módulos que contiene el pipeline de procesamiento, y como se realizó su implementación e integración. También se detallará sobre el desarrollo de la interfaz gráfica que permite utilizar esta herramienta, además de mencionar el importante proceso de modularizar y poner a disposición la herramienta.

4.1. Primeros Pasos

Para el desarrollo de la aplicación *Textarium* se optó por la utilización del lenguaje de programación *Python*[1], debido a que cuenta con una gran cantidad de bibliotecas y *frameworks* dedicados al *NLP* que facilitan el desarrollo de aplicaciones en este campo. Estas herramientas ofrecen una amplia gama de funcionalidades, como tokenización, lematización, *POS-Tagging*, reconocimiento de entidades, traducción automática, generación de texto, entre otras.

Además, *Python* es ampliamente utilizado en una variedad de campos, incluyendo *ML* y ciencia de datos. Esta integración con otras áreas y tecnologías permite combinar las capacidades del *NLP* con otras técnicas para construir soluciones más completas y avanzadas.

Dicho esto, el código fuente del sistema está disponible públicamente en la plataforma de control de versiones *GitHub*[2], junto con su documentación e instrucciones de uso en el siguiente *repositorio*. También se proporciona un *notebook* explicativo y autocontenido de *Google Colab*[3] que permite ejecutar la herramienta de forma sencilla y rápida, logrando que usuarios menos experimentados puedan levantar la herramienta y comenzar a utilizarla sin complicaciones.

En caso de utilizar el código del repositorio *GitHub*, se recomienda la instalación del siste-

[1] https://www.python.org
[2] https://github.com
[3] https://colab.research.google.com

ma de distribución *Anaconda*[4], para mejorar la gestión de librerías y crear un entorno virtual para la herramienta. *Anaconda* proporciona una solución completa y útil para el manejo de paquetes y entornos de desarrollo en *Python*, lo que facilita la instalación y actualización de las dependencias necesarias para *Textarium*. Además, al crear un ambiente virtual específico para el sistema se garantiza la coherencia y reproducibilidad de la herramienta al mantener sus dependencias aisladas, evitando conflictos entre diferentes proyectos y versiones de librerías. Esta práctica asegura que los usuarios puedan trabajar con *Textarium* en un ambiente controlado y estable, sin preocuparse por posibles interferencias con otras aplicaciones o dependencias externas.

4.2. Recursos de Terceros

4.2.1. *Hugging Face*

Como se mencionó anteriormente, *Hugging Face* es una destacada compañía de tecnología especializada en el desarrollo de herramientas y plataformas de procesamiento del lenguaje natural basadas en inteligencia artificial. Su enfoque principal radica en la creación y la puesta en disposición de modelos de *deep learning* de última generación para diversas tareas, como comprensión del lenguaje natural, generación de texto, traducción automática, entre otras.

Gracias a su visión innovadora, *Hugging Face* ha tenido un impacto positivo en la comunidad de usuarios y empresas en los últimos años al permitir el acceso gratuito a los mejores y más avanzados modelos de *NLP*. Esta estrategia ha fomentado un crecimiento significativo en la adopción de sus tecnologías.

Una de las contribuciones más destacadas de *Hugging Face* es su librería de código abierto llamada *"Transformers"*. Esta librería brinda a desarrolladores de todo el mundo la posibilidad de acceder y utilizar modelos de *NLP* de última generación preentrenados por terceros. La popularidad de *"Transformers"* ha sido impresionante y ha sido ampliamente adoptada tanto por individuos como por empresas para el desarrollo de aplicaciones basadas en *NLP*.

Además, *Hugging Face* ha creado una plataforma *web* que sirve como un extenso repositorio global de módulos de *deep learning*, especialmente enfocados en el procesamiento del lenguaje. Estos modelos son desarrollados por algunas de las empresas más reconocidas a nivel mundial, incluyendo gigantes tecnológicos como *Google*, *Microsoft* y *Amazon*.

Esta plataforma ofrece a los desarrolladores la oportunidad de compartir sus modelos, lo que fomenta un ambiente de aprendizaje y colaboración. Al estar todo alojado en un mismo punto, los usuarios pueden acceder a estos recursos, analizar su funcionamiento y aprender de las mejores prácticas.

Dentro de los modelos preentrenados que se encuentran disponibles para su uso, aparecen modelos del lenguaje como *BERT* y algunas de las primeras versiones de *GPT*. Además de algunos modelos creados a partir de *BERT*, como lo son *Multilingual BERT* (*M-BERT*) [22],

[4]https://anaconda.org

una versión multilingüe de *BERT* capaz de comprender y trabajar con 104 idiomas, o *BETO* [7], un modelo equivalente en tamaño a *BERT* pero entrenado por el DCC sobre un gran corpus de textos en español, demostrando ser altamente efectivo en tareas de *NLP* en este idioma.

También se encuentran disponibles modelos como *BART*, versiones de algunos modelos del grupo *OPUS-MT* ideales para realizar tareas de *machine translation*, e incluso algunas implementaciones especializadas de modelos de lenguaje sobre los cuales se han utilizado técnicas de *fine-tuning* para obtener mejores resultados en tareas típicas de *NLP*, volviéndolos expertos en ellas. Algunos ejemplos de esto son:

- **Modelos especializados en tareas de *Part-of-Speech Tagging*:**
  - *BETO fine-tuned on POS*[5]
  - *BETO fine-tuned on POS 16-tags*[6]
- **Modelos especializados en tareas de reconocimiento de entidades:**
  - *WikiNEuRal-Multilingual-NER*[7]
  - *BETO fine-tuned on NER*[8]
  - *BERT fine-tuned on NER*[9]
- **Modelos especializados en generación de resúmenes (*Text Summarization*):**
  - *BETO fine-tuned on MLSUM ES for summarization*[10]
  - *BART (large-sized model) fine-tuned on CNN Daily Mail*[11]
- **Modelos especializados en traducción automática (*Machine Translation*):**
  - *opus-mt-es-en*[12] (traducción automática de español a inglés)
  - *opus-mt-en-es*[13] (traducción automática de inglés a español)

4.2.2. Proyectos Independientes

Siguiendo en la temática del *machine translation*, es relevante mencionar una biblioteca *open source* de *Python* llamada *Deep Translator*[14], la cual ofrece traducciones entre diferentes idiomas de manera sencilla, flexible, gratuita e ilimitada. Su enfoque radica en la integración de múltiples traductores, lo que garantiza una amplia variedad de fuentes para

[5]https://huggingface.co/mrm8488/bert-spanish-cased-finetuned-pos
[6]https://huggingface.co/mrm8488/bert-spanish-cased-finetuned-pos-16-tags
[7]https://huggingface.co/Babelscape/wikineural-multilingual-ner
[8]https://huggingface.co/mrm8488/bert-spanish-cased-finetuned-ner
[9]https://huggingface.co/dslim/bert-base-NER
[10]https://huggingface.co/mrm8488/bert2bert_shared-spanish-finetuned-summarization
[11]https://huggingface.co/facebook/bart-large-cnn
[12]https://huggingface.co/Helsinki-NLP/opus-mt-es-en
[13]https://huggingface.co/Helsinki-NLP/opus-mt-en-es
[14]https://deep-translator.readthedocs.io/en/latest/index.html

obtener traducciones precisas, confiables y con soporte para diversos idiomas, lo que refuerza su adaptabilidad a las necesidades del mundo de la traducción.

Un punto fuerte de esta librería, es que añade niveles de abstracción a la utilización de los modelos, lo que la hace sencilla de utilizar. Además, dentro de múltiples modelos que integra es posible encontrar soporte para *Google Translate*[15], lo que la convierte excelente a la hora de querer hacer traducciones rápidas y sencillas.

Por otro lado, en plataformas como *GitHub*, también es posible encontrar proyectos *open source* creados por desarrolladores que ponen a disposición infinidades de modelos y aplicaciones relacionadas con el *NLP*. Un ejemplo de esto es *KeyBERT*[16], el cual es un proyecto enfocado en la extracción de palabras clave o *keywords* utilizando *embeddings* de *BERT*. Es una opción rápida y sencilla de utilizar para la extracción de palabras y frases clave de un documento, ya que se evita el entrenamiento de un modelo desde cero para lograr esto. Sin embargo, debido al modelo en que se basa, su rendimiento es óptimo en el idioma inglés.

4.2.3. *spaCy* y *NLTK*

Otra poderosa librería *open source* ampliamente utilizada y enfocada en *NLP* es *spaCy*[17]. Esta herramienta ofrece modelos preentrenados en varios idiomas que permiten realizar tareas como *dependency parsing*, *POS-Tagging*, *NER*, entre otras, sin tener que entrenar modelos desde cero. También es posible entrenar y personalizar modelos específicos para adaptarlos a tareas y dominios particulares, y proporciona una interfaz sencilla y bien documentada, lo que facilita su integración en proyectos de *NLP*.

La arquitectura base de un *pipeline* de procesamiento de *spaCy* puede verse la figura 4.1.

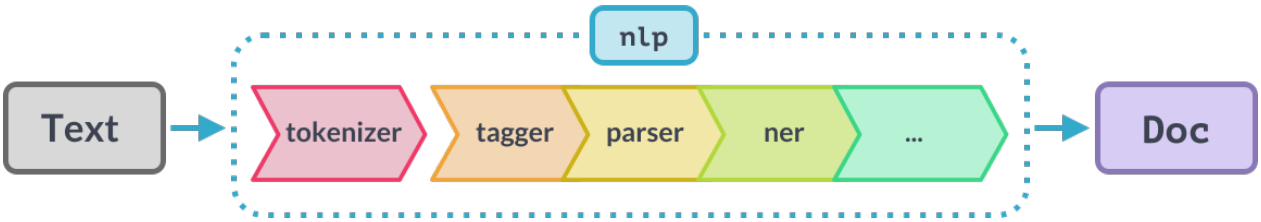

Figura 4.1: Representación del *pipeline* de procesamiento de *spaCy* [29].

A continuación se mencionarán algunos modelos preentrenados destacados

es_core_news_lg

- *Pipeline* de *NLP* en español, optimizado para ser ejecutado en CPU.
- Componentes:

[15]https://translate.google.cl/
[16]https://github.com/MaartenGr/KeyBERT
[17]https://spacy.io

- ***senter*:** Un segmentador de oraciones que divide el texto en oraciones individuales.
- ***tok2vec*:** Un componente para la tokenización y generación de vectores de palabras.
- ***attribute_ruler*:** Un componente que permite agregar atributos a los tokens en función de patrones específicos.
- ***morphologizer*:** Encargado de realizar el análisis morfológico de las palabras, es decir, determinar sus lemas y formas gramaticales.
- ***lemmatizer*:** Componente que realiza la lematización de las palabras, es decir, determina la forma, base o raíz de las palabras.
- ***parser*:** Componente encargado de realizar el análisis de dependencias sintácticas, es decir, determinar las relaciones gramaticales entre las palabras en una oración.
- ***ner*:** Reconocimiento de entidades nombradas, que identifica y clasifica entidades como nombres de personas, lugares, organizaciones, etc.

es_dep_news_trf

- *Pipeline* de *NLP* en español que utiliza el *transformer BETO* en su arquitectura.
- Componentes:
  - ***BETO*:** Se utiliza este *transformer* como componente principal, lo que puede ofrecer ventajas adicionales en términos de rendimiento y comprensión contextual.
  - ***attribute_ruler*, *morphologizer*, *lemmatizer*, *parser*:** Estos componentes desempeñan funciones similares a los del modelo *es_core_news_lg*, proporcionando la agregación de atributos, el análisis morfológico, la lematización de palabras y el análisis de dependencias sintácticas.

A diferencia de algunas otras librerías de *NLP*, *spaCy* está escrita en el lenguaje de programación *Cython*[18], lo que le permite alcanzar un alto rendimiento y velocidad, convirtiéndola en una excelente opción para aplicaciones en tiempo real o aplicaciones que utilizan grandes volúmenes de datos, destacándose por su eficiencia, precisión y facilidad de uso.

Un ejemplo de librería escrita en *Python* puro es *NLTK*[19] (*Natural Language Toolkit*), la cual se asemeja bastante a *spaCy*, pero con diferencias importantes en cuanto a su enfoque, rendimiento y características, a continuación se detallan estos puntos.

- **Enfoque y filosofía:**
  - ***spaCy*:** Está diseñada para ser rápida, eficiente y escalable. Su objetivo es ofrecer un alto rendimiento y ser adecuada para aplicaciones en tiempo real y procesamiento en grandes volúmenes de datos. *spaCy* se centra en la facilidad de uso y en proporcionar modelos preentrenados en varios idiomas para realizar tareas de *NLP* de manera rápida y sin la necesidad de entrenar modelos desde cero.

[18]https://cython.org
[19]https://www.nltk.org

  - ***NLTK*:** Se enfoca en ser una biblioteca educativa y didáctica para el procesamiento del lenguaje natural. Aunque también ofrece algunas funcionalidades avanzadas, su enfoque principal es facilitar el aprendizaje y la enseñanza de *NLP* a través de una gran cantidad de recursos lingüísticos y ejemplos.

- **Rendimiento:**
  - ***spaCy*:** Como se mencionó, está escrita en *Cython*, lo que permite un alto rendimiento, mayor velocidad y superior eficiencia en el procesamiento de texto en comparación con *NLTK*.
  - ***NLTK*:** Aunque *NLTK* es una biblioteca muy versátil y completa, su rendimiento puede ser más lento en comparación con *spaCy*, especialmente cuando se trabaja con grandes volúmenes de texto.

- **Recursos lingüísticos:**
  - ***spaCy*:** Ofrece modelos preentrenados en varios idiomas y una amplia gama de recursos lingüísticos, aunque la cantidad de modelos puede ser menor en comparación con *NLTK*.
  - ***NLTK*:** Se destaca por su amplia colección de corpus, recursos lingüísticos y herramientas para tareas educativas y de investigación en *NLP*.

4.2.4. *Wikidata*

Dentro de los objetivos del proyecto se mencionó que a la hora de la extracción de patrones, palabras recurrentes, conceptos clave, entidades e interacciones, estas serían comparadas, complementadas y validadas por una fuente de datos externa, esto con el objetivo de enriquecer el contenido entregado por los usuarios de la herramienta.

Es aquí donde entra *Wikidata*[20], una base de datos colaborativa y abierta lanzada por la Fundación Wikimedia. Su objetivo principal es recopilar y almacenar datos estructurados y enlazados de manera centralizada, para que puedan ser utilizados por diferentes proyectos de la familia *Wikimedia*, así como por otras aplicaciones y servicios en línea. En esencia, *Wikidata* actúa como un repositorio de conocimiento que almacena información en un formato legible por máquinas, lo que permite que aplicaciones puedan acceder y utilizar estos datos.

Para la integración de los datos de *Wikidata* al sistema, se pueden realizar consultas a través del lenguaje *SQL* (*Structured Query Language*) utilizando el servicio *SPARQL Endpoint* que proporciona *Wikidata*. *SPARQL* es un lenguaje de consulta diseñado específicamente para trabajar con datos en formato *RDF*, que es el formato utilizado por *Wikidata* para almacenar y representar la información.

A través de *SPARQL*, se pueden realizar consultas complejas y precisas para extraer la información necesaria de *Wikidata*. Algunos ejemplos de consultas que se pueden realizar incluyen:

[20]https://www.wikidata.org/wiki/Wikidata:Main_Page

- Obtener la descripción y propiedades asociadas a un elemento específico, como caracteristicas de un lugar, una persona o una organización.
- Realizar consultas de búsqueda para encontrar elementos que cumplan ciertas condiciones, como buscar todas las personas con una determinada profesión.
- Realizar consultas que involucren relaciones y enlaces entre diferentes elementos en *Wikidata*, lo que permite obtener información relacionada.

La integración de los datos de *Wikidata* al sistema mediante el uso de consultas *SPARQL* puede ser muy útil para enriquecer la información y aprovechar la amplia gama de datos estructurados y enlazados disponibles en esta plataforma colaborativa.

4.2.5. *Gradio*

Por último, teniendo en consideración que uno de los objetivos fundamentales del proyecto es la construcción de un *pipeline* de procesamiento que incorpore los análisis y extracciones de caracteristicas realizadas por los modelos descritos, y que se logre su integración en una interfaz sencilla de utilizar e intuitiva para los usuarios, se tiene la necesidad de buscar herramientas que sean compatibles y faciliten este desarrollo.

Es por esto que es importante hablar de *Gradio*, una librería de código abierto creada para simplificar y facilitar el desarrollo de interfaces de usuario interactivas para modelos de aprendizaje automático. Con *Gradio*, se pueden crear fácilmente aplicaciones *web* que permitan a los usuarios interactuar con los modelos de una manera sencilla y visual. Lo más destacado de esta biblioteca es que no se requiere experiencia en desarrollo *web*, lo que la hace bastante útil y accesible para aquellos que busquen compartir y demostrar sus modelos de una forma sencilla. La versatilidad de *Gradio* también se manifiesta en su capacidad para desplegar prototipos y demostraciones de manera rápida y sencilla, ya que proporciona un enlace que permite compartir las aplicaciones creadas con otras personas, sin necesidad de configurar servidores ni realizar despliegues complicados.

Al aplicar esta librería para el desarrollo de la interfaz del sistema, se logra agilizar significativamente el proceso de mostrar los avances y resultados obtenidos en el proyecto. Además de facilitar su despliegue y validación con usuarios, gracias a los enlaces proporcionados por la misma librería.

4.2.6. Limitaciones y Consideraciones

La combinación e integración de los diversos modelos, librerías y herramientas descritas anteriormente permite aprovechar sus fortalezas en la construcción de una solución bastante completa. Sin embargo, la aplicación no solo debe cumplir con su propósito, sino también ser usable por usuarios finales. Por lo tanto, es necesario encontrar un balance entre la obtención de resultados de calidad, tiempos de procesamiento y uso de los recursos computacionales con el fin de cumplir con el desarrollo integral del *MVP* de aplicación.

Dicho de otra forma, no resulta viable realizar una carga en memoria de todos los modelos mencionados y utilizarlos indiscriminadamente a lo largo del sistema, ya que esto implicaría un excesivo gasto de recursos e ineficiencia en la solución. Por ende, en el contexto del desarrollo del trabajo, se tomaron decisiones y se realizaron comparativas entre modelos con el objetivo de elegir una combinación que ofreciera el mejor equilibrio entre las categorías mencionadas.

En este sentido, aunque *Hugging Face* dispone de numerosos modelos *fine-tuned* para tareas específicas que logran excelentes desempeños, el enfoque del proyecto implica realizar varias secuencias de procesamientos encadenados con el objetivo de descomponer y analizar de la mejor manera los textos ingresados por los usuarios, generando valor y aportando contenido extra a lo largo de la solución desarrollada. Por lo tanto, a pesar de que los modelos *fine-tuned* tengan mejores rendimientos en sus respectivas especializaciones, disponer de estos modelos se traduce en la utilización de cada uno de ellos en el sistema, lo cual no es factible.

De hecho, este fue unos de los problemas que se presentaron en el desarrollo del proyecto. Al momento de realizar pruebas de concepto y trabajar sobre un *notebook* de *Google Colab*, varias veces se alcanzaron los límites de RAM asociados al entorno de ejecución.

No obstante, *spaCy* cuenta con modelos y *pipelines* de procesamiento que incluyen componentes para realizar tareas comunes de *NLP* como *dependency parsing*, *POS-Tagging*, *NER*, incluso siendo capaz de trabajar con el idioma español. Algunos modelos que fueron nombrados anteriormente y que cuentan con estas caracteristicas son *es_core_news_lg* y *es_dep_news_trf*.

Recapitulando, la arquitectura del primero presentaba componentes llamados: *senter*, *tok2vec*, *attribute_ruler*, *morphologizer*, *lemmatizer*, *parser* y *ner*, que permiten realizar tareas como *sentence segmentation*, *word tokenization*, *lemmatization*, *dependency parsing*, *POS-Tagging* y *NER*. En cambio, la arquitectura del segundo se caracterizaba por utilizar *BETO*, logrando realizar tareas similares a las anteriores, pero dejando *NER* afuera.

Por lo cual, gracias a *spaCy* es posible realizar varias tareas de *NLP* a partir de un solo modelo, alineándose con el enfoque del proyecto y siendo viable la posibilidad de realizar varias secuencias de procesamientos encadenados con el objetivo de descomponer y analizar de la mejor manera los textos ingresados por los usuarios. Ya que como se mencionó, *SpaCy* es conocido por su eficiencia y rendimiento en el procesamiento de texto, lo que contribuye a agilizar el flujo del sistema y a obtener resultados precisos en tiempo real.

Se prefirió usar el modelo *es_core_news_lg* por sobre *es_dep_news_trf*, ya que ambos cuentan con rendimientos muy similares, pero el primero tiene una componente que permite realizar tareas de *NER*, lo cual facilita aún más las cosas y permite llevar a cabo el análisis lingüístico necesario para el proyecto sin sobrepasar los límites de memoria y recursos computacionales.

Además, el "ahorro" en recursos computacionales generado por esta decisión proporciona flexibilidad para optar por la utilización de modelos específicos en caso de ser necesario. Este es el caso de *KeyBERT*, el cual se utilizará para potenciar la detección de conceptos clave. La consecuencia de esta decisión es la incorporación de modelos del grupo *OPUS-MT* (en particular, los modelos *opus-mt-es-en* y *opus-mt-en-es*) y el uso de la librería *Deep Translator*, debido a que como *KeyBERT* es un modelo que obtiene mejores resultados en

inglés es necesario traducir las entradas y salidas del modelo devuelta al español.

De las tareas típicas de *NLP* solo queda cubrir el *Text Summarization*. Para esto se optó por utilizar el modelo *BART (large-sized model) fine-tuned on CNN Daily Mail*, ya que obtuvo rendimientos superiores en la generación de resúmenes al compararlo con el modelo *BETO fine-tuned on MLSUM ES for summarization*.

Estos rendimientos superiores se ven respaldados, por un lado, al fijarse en las métricas de rendimiento reportadas en las tarjeras de información de los modelos en la página de *Hugging Face*, donde el primer modelo mencionado es el que presenta las mejores métricas. Pero también en la práctica, al momento de realizar pruebas de rendimiento en ambos modelos con *inputs* iguales, la calidad de los resúmenes generados por el modelo basado en *BART* fueron superiores, manteniendo la esencia y coherencia de los relatos originales, mientras que el basado en *BETO* fallaba más seguido. El único contra que tenía la elección del primer modelo era el hecho de que estaba entrenado en inglés, pero considerando que ya se iban a incorporar modelos de *MT*, se generó el escenario ideal para su utilización, lo cual determinó su elección finalmente.

Dicho todo esto, a continuación se describirá cómo se llevó a cabo el desarrollo de cada componente del sistema en aspectos más técnicos, mencionando cómo se incorporaron las herramientas, librerías y modelos *open source* elegidos para la construcción de la solución.

4.3. Módulo de Preprocesamiento

Recapitulando, el módulo de preprocesamiento es el primer módulo que conforma al *pipeline* desarrollado. Este interactúa directamente con los usuarios finales y se encarga de recibir, limpiar y preprocesar la entrada de texto brindada por estos. Esto se realiza con el fin de estandarizar la entrada, dejándola preparada para que las demás componentes del sistema se encarguen de realizar sus respectivas tareas. Además, este módulo aprovecha de extraer un par de características y estadísticas sencillas de ejecutar luego de un procesamiento simple.

La estandarización de la entrada se logra a través de la función `clean_text`. Esta función recibe tres argumentos: *text*, *punctuations* y *stop_words*. El primer argumento es la descripción en texto plano del relato que se está procesando, representada en formato de *string*. Los dos siguientes son parámetros por omisión que por defecto son nulos. Sin embargo, cuando adquieren valores, representan una lista de puntuaciones y una lista de palabras, respectivamente, que deben ser removidas del texto de *input*, para lograr su estandarización y dejar la entrada sin puntuaciones ni palabras vacías que no aporten valor. De todas maneras, ya sea que los parámetros por omisión sean nulos o no, se aplica la función `strip` de *Python* para remover espacios en blanco sobrantes en el texto.

Los únicos valores que son utilizados para los parámetros por omisión mencionados anteriormente son listas de puntuaciones y *stop words* en español provenientes de las librerías *spaCy* y *NLTK*, logrando cubrir en conjunto un amplio repertorio de caracteres y palabras a remover.

Dependiendo de la tarea que se esté ejecutando, como se explicó en la sección 3.2, la función de limpieza manda su salida a un submódulo auxiliar u otro, las implementaciones y detalles de cada una de las opciones son descritas a continuación.

4.3.1. Translator

La principal funcionalidad de este submódulo auxiliar es traducir las entradas de texto limpias y procesadas en diferentes etapas del *pipeline*, desde un idioma fuente a un idioma de destino. Su objetivo es actuar como un enlace entre modelos que tienen un mejor rendimiento en un idioma específico y modelos que se desempeñan mejor en otro idioma distinto, facilitando la integración y colaboración entre modelos en distintos lenguajes, y ampliando el repertorio de modelos que pueden ser utilizados a lo largo del sistema.

En específico, se pretende traducir texto del español al inglés para permitir la utilización de *KeyBERT* en la extracción de conceptos clave y de modelos especializados en la generación de resúmenes de texto (*summarizers*) en este mismo idioma, y viceversa también, para la traducción del *output* del módulo extractor de conceptos claves y el *output* del *summarizer* de vuelta al español.

Para lograr esto se usaron dos enfoques, uno con la utilización de los modelos *opus-mt-es-en* y *opus-mt-en-es* de la *Helsinki NLP* alojados en *Hugging Face*. Y otra implementación que usa el modelo de *Google Translate* gracias a la librería *Deep Translator*.

El desarrollo de la primera variante se realizó haciendo uso de la librería *Transformers* de *Hugging Face*. Su uso es bastante directo por medio de las clases *AutoTokenizer* y *AutoModelForSeq2SeqLM*. Con ella se pueden realizar los dos procesos principales de cualquier modelo basado en *transformers*: *tokenization* y el procesamiento. La primera clase permite transformar el texto plano a *word embeddings*, mientras que la segunda emplea estos *embeddings* para obtener el output deseado. Por lo cual se implementó la función `llm_translator`, que implementa la lógica descrita.

Mientras que el desarrollo del otro enfoque es más directo haciendo uso métodos de *Deep Translator*, como se mencionó anteriormente, el beneficio de esta librería es aportar niveles de abstracción extra en la traducción de texto, siendo particularmente eficiente y rápida para entradas pequeñas. Este es el caso de los conceptos claves detectados por el modelo *KeyBERT*, por lo cual su combinación se ve potenciada.

4.3.2. Summarizer

Al igual que para la implementación de la función que hace uso del modelo de lenguaje enfocado en traducir, se implementó una función `llm_summarizer`, mediante las clases *AutoTokenizer* y *AutoModelForSeq2SeqLM*. El modelo del lenguaje utilizado por detrás es la versión *fine-tunned* de *BART* especializada en resúmenes.

4.3.3. StraightTextStats

Este submódulo es la tercera opción de salida de módulo de preprocesamiento, como se mencionó anteriormente, se encarga de extraer estadísticas y métricas de fácil ejecución sobre entradas de texto a penas procesadas.

Se genera una nube de palabras completa del texto ingresado utilizando la librería *word-cloud* de *Python*. Por otro lado, también se extraen las palabras más frecuentes en el texto de entrada. Para lograr esto primero se hace uso del *lemmatizer* de *spaCy* y una vez que las palabras estén en su raíz, se procede a contar la cantidad de repeticiones de cada una apoyándose de la librería *collections*

4.4. Módulo de Extracción de Conceptos Claves

Como ya se ha mencionado, utiliza *KeyBERT* para la detección de entidades, traduciendo las salidas de vuelta al español cuando se muestra la información a los usuarios.

4.5. Módulo de Detección de Entidades

El módulo de detección de entidades tiene un comportamiento un poco más complejo. En él se realiza una división en extractos del texto original, estos extractos son analizados por separados mediante el modelo mencionado de la librería *spaCy*. El objetivo de realizar esto es que se logra una mayor cobertura y oportunidades para la detección de las entidades, analizar varios fragmentos envés de uno solo demostró entregar mejores resultados. Además, este módulo hace uso de los conceptos claves detectados en el paso anterior, con el objetivo de reforzar aún más la consolidación

4.5.1. Desambiguación de Entidades

Una vez detectada la entidad, se realizan comparaciones con la base de datos de *Wikidata* esperando una coincidencia. Esto se hace con el objetivo de desambiguar la detección de la entidad y entregar una representación consolidada por los datos de esta plataforma.

Una vez identificada y desambiguada la entidad, también se procede a buscar entidades similares que compartan propiedades en común con la representación escogida. Las consultas hacia la *API* de *Wikidata* se realizan haciendo uso de la librería *requests* de *Python*, además como se mencionó se utiliza *SPARQL* para buscar eficientemente la data en las bases indexadas de *Wikidata*.

4.6. Módulo de Creación de Ilustraciones

El módulo de creación de ilustraciones hace uso del modelo de difusión *stable diffusion* 1.5. para la generación de las ilustraciones. Cada extracto generado en los pasos anteriores es pasado como *input* al modelo, logrando de esta forma obtener representaciones del relato en forma linear, logrando mostrar una historia coherente con las imágenes.

4.7. Desarrollo de la Interfaz

En cuanto al desarrollo de la interfaz de la aplicación, se siguió tomando un enfoque modular, en la cual se hizo uso de la librería *Gradio* y sus componentes *Blocks* para construir interfaces independientes por módulo.

La interfaz general de la aplicación es intuitiva, compartiendo una misma caja de *input* para los diferentes módulos, mientras que las respuestas entregadas por los modelos fueron procesadas para ser envueltas en componentes *HTML* y así obtener una interfaz más producida a la vez que permitía visualizar los *outputs* de manera más organizada.

Por un lado, la interfaz que contiene al módulo de preprocesamiento solo expone los resultados del *pipeline* al momento de correr el procesamiento. Mientras que la interfaz del módulo de detección de conceptos clave ofrece un poco más de configuración. En ella se exponen parámetros ajustables por los usuarios que definen el largo en cantidad de palabras de los extractos que se realizarán a partir del texto original, con el fin de facilitar el procesamiento de la entrada, y la cantidad de conceptos claves que se quieren buscar dentro del relato.

La interfaz del módulo de reconocimiento de entidades es más compleja, presenta varios parámetros ajustables que permiten elegir qué categorías serán relevantes al momento de buscar entidades similares, además de limitar aspectos como la cantidad de entidades que se buscarán y el tiempo de cada petición.

Finalmente, la interfaz del módulo de ilustración es la más completa, presenta componentes deslizables para las imágenes generadas, además expone parámetros para definir que estilo de ilustración será generada por el modelo. Esto se logra complementando los extractos que serán ilustrados con técnicas de *prompt engineering* que aporten características y palabras relacionadas con el estilo en particular.

Capítulo 5

Evaluación y Validación de Resultados

En este capítulo se proporciona un ejemplo guiado en que se hace uso del sistema de análisis y visualización de textos *Textarium*. Además, se expondrá como se realizó el proceso de validación con usuarios, se revisarán las respuestas al formulario de desempeño de la aplicación y de los comentarios retroalimentativos entregados por ellos.

5.1. Ejemplo Guiado

Con el fin de mostrar los resultados obtenidos por el *pipeline* de procesamiento y de mostrar su completa integración en una interfaz intuitiva, generando un sistema integral listo para ser utilizado por usuarios finales en forma de aplicación, se realizará un ejemplo guiado en que se hará uso de la herramienta simulando el comportamiento de un usuario.

Para lograr esto, se hizo un despliegue de la aplicación en forma local, se ingresó una descripción de un relato en el sistema, se ejecutó la herramienta módulo por módulo y se visualizaron los resultados obtenidos.

A continuación se procederá a explicar como es la vista al momento de hacer despliegue de la aplicación.

5.1.1. Inicialización del Sistema

Al desplegar la aplicación, se tiene una vista inicial en la cual es posible observar una gran caja de texto que invita a los usuarios a proporcionar una descripción de un relato (Ver Figura 5.1).

Además, se pueden visualizar pestañas con nombres que hacen referencia a los distintos módulos descritos en las secciones de diseño de la solución e implementación de la solución. Estos cuatro son:

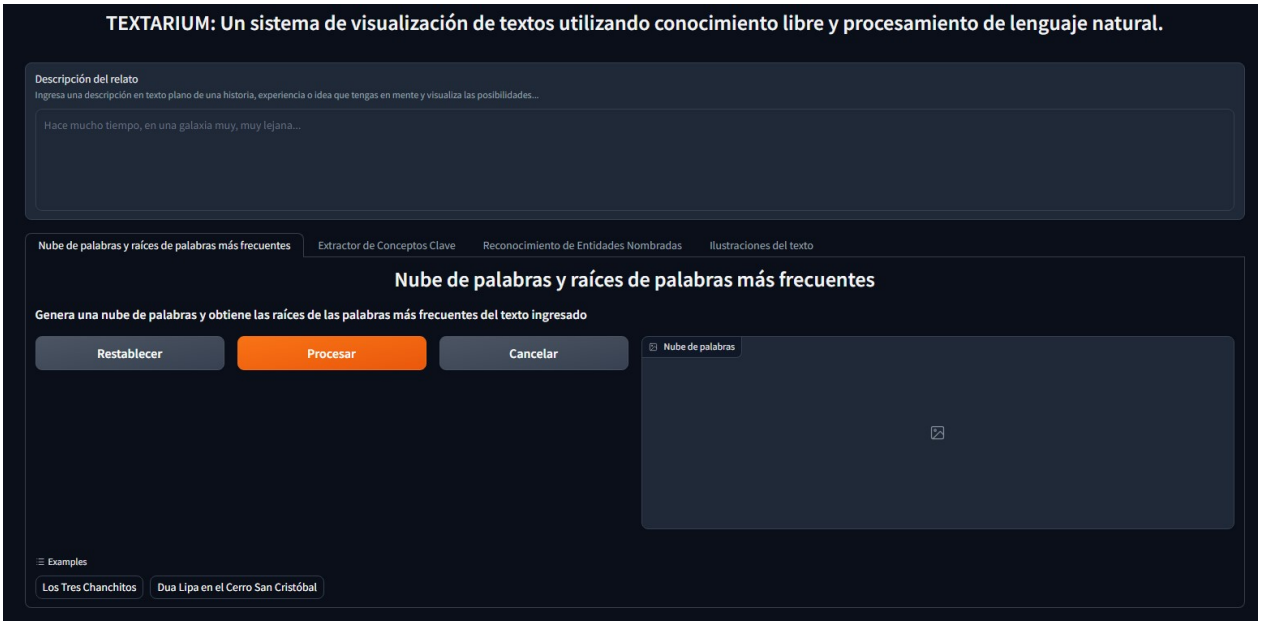

Figura 5.1: Primera vista a la aplicación.

- Nube de palabras y raíces de palabras más frecuentes
- Extractor de Conceptos Clave
- Reconocimiento de Entidades Nombradas
- Ilustraciones del texto

También es posible apreciar algunos botones que indican cómo utilizar la aplicación e incluso algunos que brindan textos de ejemplo para probar la aplicación.

5.1.2. Utilización del Sistema

Para continuar con el ejemplo guiado, se procesa un texto sugerido por la aplicación y se muestran los resultados generados. El texto elegido corresponde a una breve historia que trata sobre la visita de la artista Dua Lipa a Chile, en la cual subió el cerro San Cristóbal y probó el mote con huesillo.

Dada esta historia como entrada, los resultados arrojados por la pestaña "Nube de palabras y raíces de palabras más frecuentes" corresponden a los resultados generados por el módulo de preprocesamiento, los cuales básicamente son visualizaciones de los *outputs* del submódulo auxiliar *StraightTextStats*, que consisten en la generación de una nube de palabras y una lista de las raíces de palabras más frecuentes y se muestran en la figura 5.2 y en la figura 5.3.

Figura 5.2: Nube de palabras.

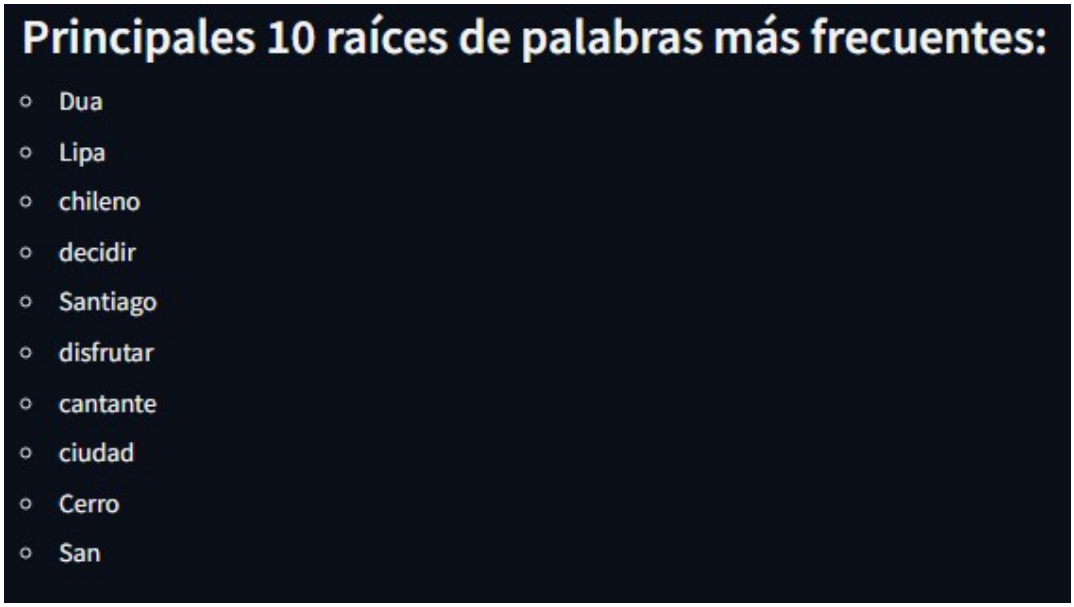

Figura 5.3: Raíces de palabras más frecuentes.

Al cambiarse a la pestaña “Extractor de Conceptos Clave” es posible observar la interfaz representada en la figura 5.4, en ella se puede ver como se permite la configuración de los parámetros que definen el tamaño de los extractos y el número de conceptos a detectar, los resultados al presionar el botón procesar pueden verse en la figura 5.5

Figura 5.4: Interfaz de la pestaña “Extractor de Conceptos Clave”.

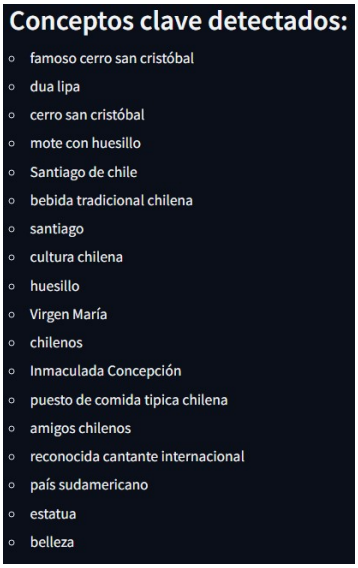

Figura 5.5: Conceptos Clave.

Al hacer *click* en la pestaña "Extractor de Entidades" es posible observar la interfaz representada en la figura 5.6, en ella se puede ver como se permite la configuración de los parámetros que definen las propiedades y cantidad de categorías a considerar al momento de realizar la búsqueda de entidades similares, además de limitar la cantidad de resultados y segundos que demoran las consultas. Los resultados al presionar el botón procesar pueden verse en la figura 5.7 y la figura 5.8

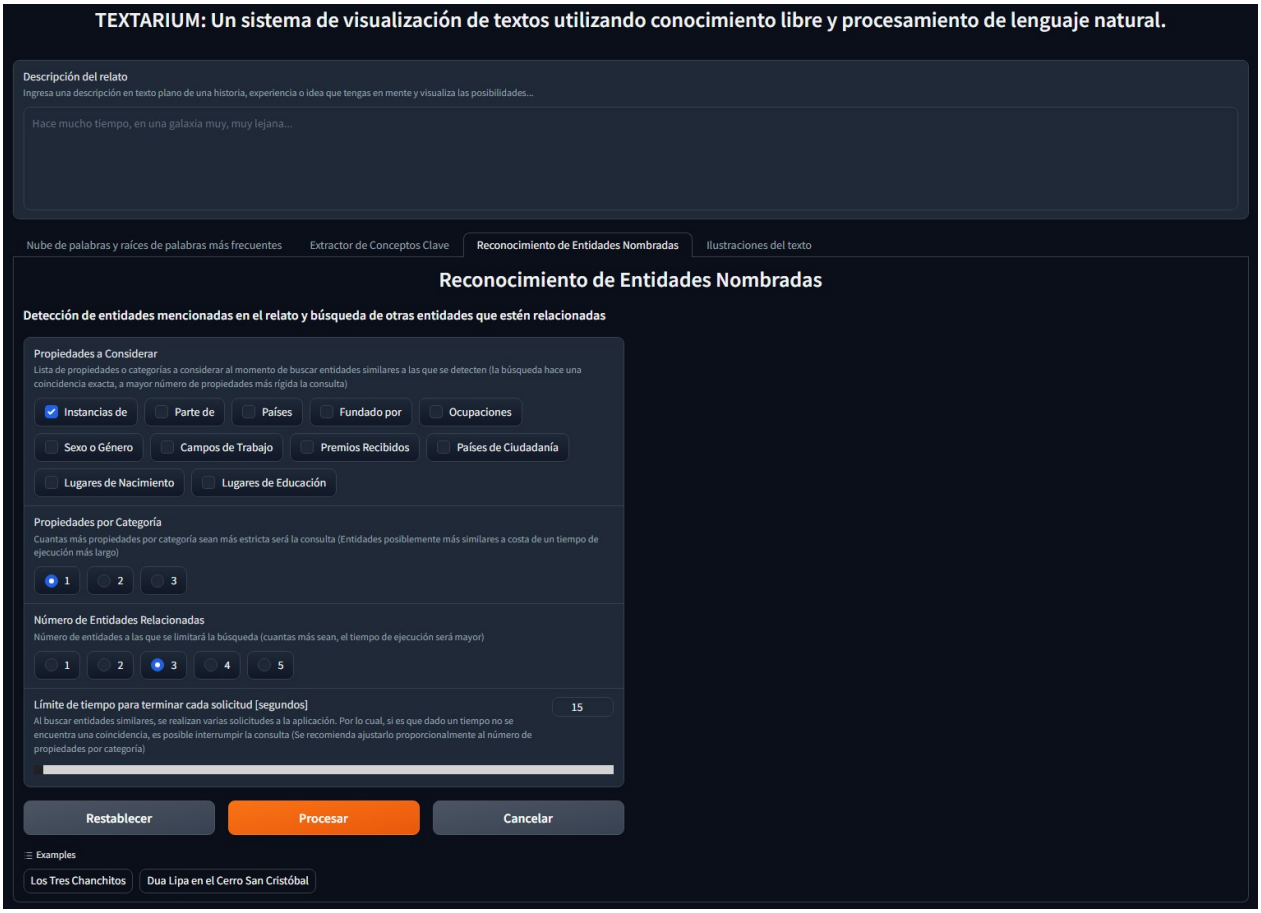

Figura 5.6: Interfaz de la pestaña "Extractor de Entidades".

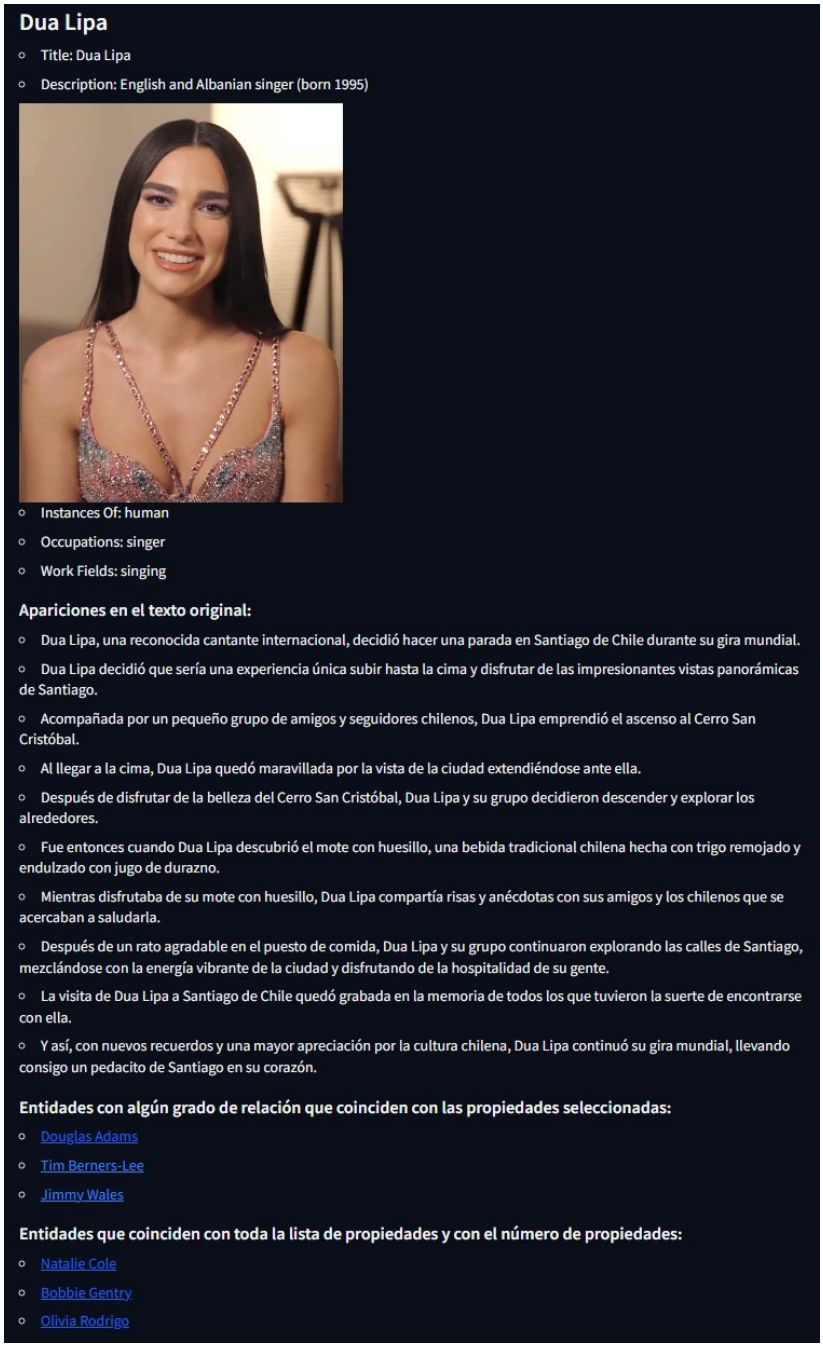

Figura 5.7: Ejemplo de Entidad Detectada.

Entidades Detectadas:

Cerro San Cristóbal

- Title: Cerro San Cristóbal
- Description: Santiago

- Instances Of: mountain
- Countries: Chile

Apariciones en el texto original:

- Uno de los lugares que le llamó especialmente la atención fue el famoso Cerro San Cristóbal, una imponente colina ubicada en el corazón de la ciudad.
- Acompañada por un pequeño grupo de amigos y seguidores chilenos, Dua Lipa emprendió el ascenso al Cerro San Cristóbal.
- Después de disfrutar de la belleza del Cerro San Cristóbal, Dua Lipa y su grupo decidieron descender y explorar los alrededores.
- La cantante se llevó consigo no solo la experiencia de subir al Cerro San Cristóbal y disfrutar del mote con huesillo, sino también el amor y el cariño de los chilenos que la recibieron con los brazos abiertos.

Entidades con algún grado de relación que coinciden con las propiedades seleccionadas:

- Grivola
- Gran Paradiso
- Cervino

Entidades que coinciden con toda la lista de propiedades y con el número de propiedades:

- Volcán Lonquimay
- Volcán Llullaillaco
- Monte Fitz Roy

Figura 5.8: Ejemplo de una segunda Entidad Detectada.

Por último, al cambiarse a la pestaña “Ilustraciones del texto” es posible observar la interfaz representada en la figura 5.9, en ella se puede ver como se permite la configuración de los parámetros que definen el estilo de las ilustraciones, pudiendo elegir categorías como fantasía, realista, entre otras. Además de poder elegir las dimensiones de las matrices de ilustraciones de los extractos generados en pasos anteriores, logrando de esta manera elegir la cantidad total de imágenes que serán generadas. Los resultados al presionar el botón procesar pueden verse en la figura 5.10

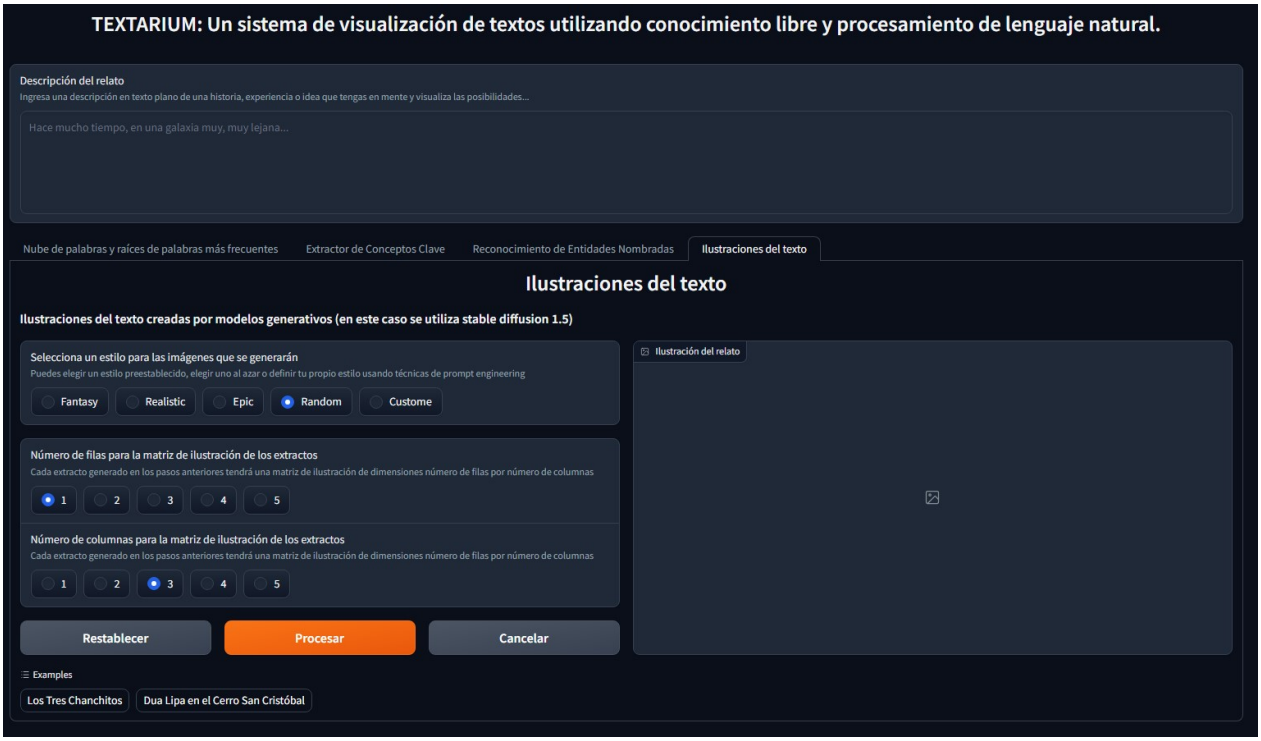

Figura 5.9: Interfaz de la pestaña “Ilustraciones del texto”.

Figura 5.10: Dua Lipa en el Cerro San Cristóbal.

5.2. Validación con Usuarios

Luego de realizar una prueba local y con el objetivo de buscar una validación de fuentes externas, se realizó un proceso de validación con un conjunto acotado de usuarios. A estos se les solicitó utilizar la herramienta y que posteriormente rellenaran un formulario orientado en la validación de rendimiento, usabilidad y calidad de resultados entregados por la herramienta de análisis y visualización de textos *Textarium*.

5.2.1. Metodología

Se reclutó a un grupo acotado de 6 personas para participar en un estudio de usuarios cuyo objetivo era validar y medir el desempeño del sistema desarrollado. Se les pidió a los usuarios que utilizaran la aplicación y que posteriormente rellenaran una encuesta de evaluación de rendimiento de la aplicación. Los usuarios sometidos al estudio fueron escogidos al azar dentro de un universo de posibles candidatos, pero cuidando que la proporción entre la gente que tenía conocimientos relacionados con la computación, se mantuviera similar a la proporción de gente que no manejaba estos conceptos.

Para lograr realizar las pruebas de usuario, se levantó la aplicación de manera local utilizando los recursos computacionales disponibles localmente. Como se mencionó anteriormente, la librería *Gradio* se destaca por su capacidad para desplegar prototipos y demostraciones de manera rápida y sencilla, ya que al momento de levantar una aplicación en modo público se genera automáticamente un enlace que permite compartir las aplicaciones creadas con otras personas, sin necesidad de configurar servidores ni realizar despliegues complicados. Esto facilitó la realización de pruebas con usuarios debido a que solo hizo falta hacer envío del enlace generado por *Gradio* para poder utilizar la aplicación desde una máquina externa a la que estaba ejecutando la aplicación.

Por el lado del cuestionario, se desarrolló un documento de *Google Forms* que consistió en el encadenamiento de 6 secciones. La primera sección se utilizó para dar contexto sobre la aplicación y el proceso que se iba a realizar, aquí se dieron indicaciones de que las respuestas generadas serían totalmente anónimas y no se guardaría registro de los textos procesados al momento de realizar las pruebas. Las siguientes 4 secciones se destinaron para evaluar el desempeño de cada módulo por separado, las preguntas realizadas fueron:

- *¿La calidad de los resultados entregados por el módulo fue buena?*
- *¿La interfaz te resultó intuitiva y fácil de utilizar?*
- *¿El tiempo de ejecución fue el esperado de acuerdo a las opciones elegidas?*

Además de una pregunta opcional que permitió que los usuarios ingresaran comentarios con respecto al módulo en específico que estaba siendo evaluado.

La sección final fue destinada a la evaluación general del sistema. En ella se realizaron dos preguntas opcionales con el objetivo de recopilar sugerencias.

La primera pregunta se centró en obtener información acerca de posibles cambios sugeridos para mejorar la interfaz o los análisis actuales que se llevan a cabo al utilizar la aplicación. Mientras que la segunda pregunta se enfocó en recolectar información con respecto a nuevas funcionalidades del sistema o módulos adicionales que fueran de interés para los usuarios.

5.2.2. Resultados

De la validación realizada con usuarios se logró recopilar información tanto de la experiencia de usuario, como del rendimiento del sistema. A continuación se expondrán los resultados obtenidos, junto con los comentarios más relevantes hechos. Cabe mencionar que en la siguiente sección se realizará una discusión global con respecto al sistema y a los comentarios entregados por los usuarios.

Módulo de Preprocesamiento

Los *outputs* de este módulo son la nube de palabras del texto y las palabras más frecuentes. Se destacó la calidad de los resultados obtenidos, la sencillez de la interfaz y el bajo tiempo de ejecución.

Los principales comentarios guardan relación con que en algunos casos las palabras presentadas en la nube eran sub- palabras que no aparecían realmente en el relato, ya que formaban parte de una palabra compuesta. Por ejemplo, dado un cuento ambientado en la ciudad de *Nueva York* la nube destacó solo la palabra *York* y no el concepto que hay detrás.

Otro comentario que se recibió fue que para relatos largos la nube de palabras entregaba demasiados resultados, lo cual podría ser solucionado devolviendo como *output* menos palabras y que fueran las realmente importantes.

Módulo de Extracción de Conceptos Clave

En este caso se destacó la calidad de los resultados y, por sobre todo, el bajo tiempo de ejecución del módulo, pero se experimentó cierta dificultad en el uso de la interfaz.

Los comentarios realizados se relacionan con que no fue intuitivo para qué servía la elección del tamaño de los extractos, ni en que influía el hecho de modificar el valor por defecto. Se sugirió incluir en la interfaz el largo actual del relato ingresado, con el fin de facilitar la elección de tamaño de los extractos y el número de conceptos a detectar, o incluso una funcionalidad que en forma automática optimice los parámetros (largo extractos y cantidad conceptos) en base al largo del texto original.

Módulo de Detección de Entidades

Este módulo generó opiniones divididas, se generaron numerosos comentarios con respecto a la usabilidad de la interfaz y al lenguaje técnico utilizado en ella. Se sugirió explicar de mejor manera en que consistía el reconocimiento de entidades, y las categorías de las propiedades elegidas al momento de realizar la búsqueda de una entidad similar.

Sin embargo, también se destacaron los resultados obtenidos, esto da a entender que a pesar de que hubo dificultad en un principio para la utilización de la funcionalidad, una vez entregado el resultado en formato *HTML* hizo sentido y se logró comprender el propósito del módulo. Una sugerencia hecha con respecto a los resultados entregados al momento de comparar una entidad detectada en el texto con datos de *Wikidata* fue que se podrían intentar vincular más características a la entidad detectada, evitando de esta manera "falsos positivos" al momento de consolidar la información con *Wikidata*.

Algunas sugerencias extras recayeron en problemáticas relacionadas con el tiempo de ejecución. Se mencionó que sería bueno que envés de mostrar los segundos que lleva corriendo el módulo, podría añadirse una barra de progreso que indicara en que punto del análisis se encuentra.

Módulo de Creación de Ilustraciones

La evaluación de este módulo fue bastante buena. Se destacaron la calidad de los resultados obtenidos y algunos puntos de la interfaz, como la capacidad de elección de diferentes estilos de ilustración.

Los comentarios que se realizaron tienen relación con que no queda muy claro que es una "matriz de ilustración" y la relación entre la cantidad de imágenes generadas con los parámetros utilizados. Se recomendó reducir el lenguaje técnico, explicar de mejor manera la cantidad de fotos finales que se ilustrarán o incluso que el usuario final elija solamente la cantidad de imágenes y no los parámetros por fila y columna.

También se sugirió que sería interesante que al elegir un estilo de ilustración *random* salieran imágenes con diferentes estilos, en vez de todas las imágenes con un único estilo elegido al azar.

Capítulo 6

Discusión

6.1. Implicancias

El trabajo que se realizó en este proyecto, enfocado en la creación del sistema *Textarium* que permite el análisis y la visualización de textos utilizando conocimiento libre y procesamiento del lenguaje natural, trae consigo una serie de implicaciones teóricas y prácticas.

Por un lado, una implicancia netamente teórica fue la validación de factibilidad de la creación de una herramienta que cumpla con los objetivos mencionados anteriormente, realizando pruebas de concepto, integraciones y comparaciones entre diversas herramientas, librerías y modelos relacionados con el campo del *deep learning*.

Pero no limitándose exclusivamente a esto, ya que se continuó con una implementación práctica que se apoyó en recursos como modelos de *NLP*, modelos de difusión, fuentes de datos libres como *Wikidata*, librerías *open source*, entre otros, logrando el desarrollo de una aplicación integral y utilizable por usuarios finales.

Dentro de las características que destacan a esta herramienta es que permite el ingreso de fragmentos de texto para que sean procesados, analizados y caracterizados por medio de un *pipeline* de procesamiento capaz de generar contenido adicional que sea relevante y útil para los usuarios, dotándola de aplicaciones prácticas como por ejemplo servir de fuente de inspiración para la superación de problemas relacionados con los procesos creativos y de redacción.

Debido a como se enfocó el desarrollo de la herramienta, esta cuenta con módulos aislados y perfectibles, lo que otorga un potencial de mejora gigantesco en los resultados entregados por el sistema. Dada una tarea cualquiera que se quiera mejorar, basta con la modificación del módulo correspondiente, adaptándolo para la incorporación de alguna herramienta que se desempeñe mejor o incluso agregándole funcionalidades nuevas.

Todo apunta a que las características del sistema brindan las herramientas necesarias para que las personas que buscan nuevos niveles de excelencia en sus creaciones puedan perfeccionar sus trabajos, potenciando la industria creativa como nunca antes.

6.2. Limitaciones

A pesar de que *Textarium* ha cumplido con el objetivo de ser una herramienta útil para el análisis y visualización de texto, es posible identificar limitaciones que no permiten explotar su potencial.

Primero que todo, el estado actual en que se encuentra *Textarium* es el de un sistema bastante limitado y que no deja de ser un *MVP*. Los rendimientos y resultados presentados por la herramienta son bastante perfectibles y se encuentran lejos de ser los óptimos.

Con el objetivo de lograr implementar una solución integral y que fuera capaz de realizar las secuencias de análisis propuestas, se descuidó durante el proceso de desarrollo la calidad del *software* que se estaba creando. Dejando deuda técnica en el sistema y provocando algunos fallos de lógica en la implementación de algunos módulos, como la constante utilización de modelos de *MT* o como falta de explicaciones más sencillas en su interfaz. Esto último se vio reforzado con los comentarios y resultados de la validación realizada con usuarios.

Otra limitación puede verse en su componente de ilustración semántica. Actualmente, se utiliza el modelo *stable diffusion* 1.5 para la generación de imágenes, este modelo es capaz de generar buenas imágenes, pero requiere de un buen dominio en técnicas de *prompt engineering* para la liberación de todo su potencial artístico.

6.3. Trabajo Futuro

El trabajo futuro consiste principalmente en mejorar cada uno de los módulos para así obtener una herramienta con un desempeño total, mejor y mayor facilidad de uso por parte de los usuarios. Para lograr esto se pueden tomar varios caminos.

Primero, es posible extender el trabajo realizado agregando nuevos análisis y métricas a los módulos que correspondan, o incluso agregar nuevos módulos, como por ejemplo uno que continúe el trabajo realizado con las técnicas de *POS-Tagging*.

Por otra parte, se podrían implementar y probar nuevas técnicas de *NLP*, así como *LLMs* más potentes o modelos generativos que produzcan imágenes de mayor calidad, todo con el objetivo de mejorar los resultados obtenidos.

Una mejora directa que se podría hacer es utilizar directamente un modelo similar a *GPT-3* o *GPT-4*, los cuales pueden facilitar bastante el trabajo mediante resúmenes más exactos y descripciones más correctas de los relatos. Incluso se podrían mejorar enormemente la calidad de las imágenes obtenidas delegando las técnicas de *prompt engineering* a estos *LLMs*.

También hay que mencionar todas las sugerencias brindadas por los usuarios en la encuesta de desempeño, donde también se expusieron soluciones a estas problemáticas y recaen como posibles implementaciones futuras. Además, se pueden extender las pruebas con usuarios a mayor escala y poner a prueba la aplicación bajo el ojo crítico de escritores reales.

Capítulo 7

Conclusión

Tras finalizar el desarrollo de la herramienta de análisis y visualización de textos, se concluye lo siguiente:

Los resultados obtenidos a lo largo del proceso de desarrollo del sistema *Textarium* fueron satisfactorios. Se logró diseñar, implementar e integrar una herramienta realizada con software totalmente libre, que es capaz de procesar, analizar y comprender relatos de cualquier tipo. Además de brindar contenido nuevo que genera aporte y que represente un apoyo para los usuarios que redactan estos relatos.

Cada uno de los módulos desarrollados logró su objetivo, obteniendo resultados con alto rendimiento y satisfacción para los usuarios. Por lo cual, también se comprobó que es posible la creación de un pipeline de procesamiento que extraiga características clave de un texto, y que sea capaz de desglosar el lado sintáctico y semántico de él. Además, se pueden integrar y combinar distintos modelos de lenguajes y modelos generativos para fortalecer el pipeline y que se logre un mejor desempeño en el procesamiento y análisis que se busque realizar.

El desarrollo de una interfaz intuitiva y autoexplicativa fue un acierto para la utilización de la aplicación, permitiendo que la herramienta cumpla su objetivo al momento de querer ocuparla de apoyo, como fuente de inspiración o como desbloqueo creativo. Además, de que los *outputs* generados por los modelos de difusión que permiten visualizar como luce una escena descrita en texto plano aporta de gran manera en la comprensión de un texto.

La retroalimentación dada por los usuarios fue principalmente positiva, y se caracterizó por estar acompañada de una gran cantidad de mejoras realizables para futuras iteraciones

De esta forma, los grandes modelos de lenguaje y los modelos generativos pueden apoyar el trabajo creativo humano, al reducir tiempos y aportar detalles u otras perspectivas que quizás no se tenían en consideración.

Bibliografía

[1] Huggingface course. how do transformers work? https://huggingface.co/course/chapter1/4, Año de acceso al curso.

[2] Stability AI. Stable diffusion launch announcement. *Stability AI Blog*, 2022.

[3] Ewelina P. Bogucka, Beatrice A. Aseniero, Luca M. Aiello, and Daniele Quercia. The dreamcatcher: Interactive storytelling of dreams. *IEEE Computer Graphics and Applications*, 41(3):105–112, May-June 2021.

[4] Sarah Brown and David Lee. Word embeddings: An overview and applications in natural language processing. *Journal of Natural Language Processing*, 15(2):120–135, 2023.

[5] S. Carlos. Intro al natural language processing (nlp) 2 - ¿qué es un embedding? 2020.

[6] Loyola University Chicago. How to understand a prompt, 2023.

[7] DCC Uchile. Spanish word embeddings. https://github.com/dccuchile/spanish-word-embeddings, 2019.

[8] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4171–4186, 2018.

[9] Etecé Equipo editorial. Arte rupestre. *Enciclopedia Humanidades*, 2023.

[10] Anna Fogli, Luca Aiello, and Daniele Quercia. Our dreams, our selves: automatic analysis of dream reports. *Royal Society Open Science*, 7(192080), 2020.

[11] Ainur Gainetdinov. Diffusion models vs. gans vs. vaes: Comparison of deep generative models. *Towards AI*, 2023.

[12] María Gómez and Carlos Rodríguez. Unsupervised models for natural language processing: A comprehensive review. *Computational Linguistics*, 35(4):200–220, 2023.

[13] Ananda Hange. Target prediction using single-layer perceptron and multilayer perceptron. *Nerd For Tech*, 2021.

[14] Jeremy Howard and Sebastian Ruder. Universal language model fine-tuning for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 328–339, 2018.

[15] Mikhail V. Koroteev. Bert: A review of applications in natural language processing and understanding. *arXiv preprint arXiv:2103.11943*, 2021.

[16] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdel-rahman Mohamed, Omer Levy, and Veselin Stoyanov. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 1–11, 2019.

[17] Monkey Learn. Natural language processing (nlp): 7 key techniques. https://monkeylearn.com/blog/natural-language-processing-techniques/, 2022.

[18] K. Mote. Natural language processing - a survey. *Computing Research Repository*, 2012.

[19] Aleksa Nikolić. What is stable diffusion and how does it work? *Vega IT*, 2023.

[20] Layla Oesper, Daniele Merico, Ruth Isserlin, and Gary D. Bader. Wordcloud: a cytoscape plugin to create a visual semantic summary of networks. *Source Code for Biology and Medicine*, 6(1):7, 2011.

[21] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, ..., and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[22] Telmo Pires, Eva Schlinger, Dan Garrette, and Benjamin Wing. How multilingual is multilingual bert? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 4996–5001, 2019.

[23] Lawrence R. Rabiner. A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.

[24] Sebastian Ruder. An overview of multi-task learning in deep neural networks. *arXiv preprint arXiv:1706.05098*, 2017.

[25] David E. Rumelhart, Geoffrey E. Hinton, and Ronald J. Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, 1986.

[26] Syariful Shamsudin. *The Development of Neural Network Based System Identification and Adaptive Flight Control for an Autonomous Helicopter System.* PhD thesis, 08 2013.

[27] John Smith and Emily Johnson. One hot encoding: A comprehensive study. *Journal of Machine Learning*, 25(3):45–60, 2022.

[28] Michael Smith and Emily Johnson. Large language models: A survey and analysis. *Journal of Artificial Intelligence Research*, 25(3):100–120, 2023.

[29] Spacy Team. Language processing pipelines. *Spacy*, 2023.

[30] Jörg Tiedemann and Santhosh Thottingal. OPUS-MT – building open translation services for the world. In *Proceedings of the 22nd Annual Conference of the European Association for Machine Translation*, pages 479–480, Lisboa, Portugal, November 2020. European Association for Machine Translation.

[31] Turing. How does natural language processing function in ai? *Turing*, 2023.

[32] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems 30 (NIPS 2017)*, pages 5998–6008, 2017.