



Universidad
Nacional
de Loja

Universidad Nacional de Loja

Facultad de la Energía, las Industrias y los Recursos
Naturales No Renovables

Carrera de Computación

Prototipo de Software Anti-plagio para la Carrera de Ingeniería en Computación de la Universidad Nacional de Loja, utilizando Redes

Trabajo de Integración Curricular previa
a la obtención del título de Ingeniero en
Ciencias de la Computación

AUTOR:

Santiago Alexander Román Silva

Jaime Oswaldo Paqui Medina

DIRECTOR:

Ing. Óscar Cumbicus Pineda Ortega, Mg. Sc

Loja - Ecuador
2023

Certificación de director

Autoría

Nosotros, **Jaime Oswaldo Paqui Medina y Santiago Alexander Román Silva**, declaramos ser autores del presente Trabajo de Integración Curricular y eximimos expresamente a la Universidad Nacional de Loja y a sus representantes jurídicos, de posibles reclamos y acciones legales, por el contenido del mismo. Adicionalmente aceptamos y autorizamos a la Universidad Nacional de Loja la publicación de mi Trabajo de Integración Curricular, en el Repositorio Digital Institucional – Biblioteca Virtual.

Firma:

Cédula de identidad: 1900549170

Fecha:

Correo electrónico: jaime.paqui@unl.edu.ec

Teléfono: 0985606783

Firma:

Cédula de identidad: 1106041989

Fecha:

Correo electrónico: santiago.roman@unl.edu.ec

Teléfono: 0990381254

Carta de autorización del estudiante

Nosotros, **Jaime Oswaldo Paqui Medina y Santiago Alexander Román Silva**, declaramos ser autores del Trabajo de Integración Curricular titulado “Prototipo de Software Anti-plagio para la Carrera de **Ingeniería en Computación** de la Universidad Nacional de Loja, utilizando Redes” como requisito para optar por el título de Ingenieros en Computación, autorizamos al Sistema Bibliotecario de la Universidad Nacional de Loja para que, con fines académicos, muestre al mundo la producción intelectual de la Universidad, a través de la visibilidad de su contenido en el Repositorio Institucional. Los usuarios pueden consultar el contenido de este trabajo en el Repositorio Institucional, en las redes de información del país y del exterior, con los cuales tenga convenio la Universidad.

La Universidad Nacional de Loja, no se responsabiliza por el plagio o copia del trabajo de titulación que realice un tercero.

Para constancia de esta autorización, en la ciudad de Loja, a los cuatro días del mes de mayo de dos mil veintidós.

Firma:

Autor: Jaime Oswaldo Paqui Medina

Cédula de identidad: 1900549170

Fecha:

Correo electrónico:
jaime.paqui@unl.edu.ec

Teléfono: 0985606783

Firma:

Autor: Santiago Alexander Román Silva

Cédula de identidad: 1900549170

Fecha:

Correo electrónico:
santiago.roman@unl.edu.ec

Teléfono: 0990381254

DATOS COMPLEMENTARIOS:

Director del Trabajo de Titulación: Ing. Óscar Cumbicus Pineda Ortega, Mg. Sc..

Dedicatoria

Agradecimiento

Índice de Contenidos

Certificación de director.....	ii
Autoría.....	iii
Carta de autorización del estudiante.....	iv
Dedicatoria	v
Agradecimiento.....	vi
Índice de Contenidos.....	vii
Índice de Tablas	1
Índice de Figuras	1
Índice de Anexos	1
1. Título.....	2
2. Resumen	2
3. Introducción.....	2
4. Marco Teórico	2
4.1. Plagio	2
4.1.1. Tipos de Plagio	2
4.1.1.1. Tipos de plagio por su forma	2
4.1.1.2. Tipos de plagio mediante métodos	3
4.1.1.3. Tipos de plagio por su propósito	3
4.2. Formulario de Google	4
4.3. Python	4
4.4. Django	5
4.5. Redes Neuronales	5
4.5.1. Modelo estándar de neurona artificial.....	5
4.5.2. Arquitectura.....	7
4.5.3. Ventajas de las Redes Neuronales	8
4.5.3.1. Aprendizaje Adaptativo.....	8
4.5.3.2. Capacidad de generalización.....	8
4.5.3.3. Procesamiento paralelo	9
4.5.3.4. Adaptabilidad y flexibilidad.....	9
4.5.3.5. Manejo de datos complejos	9
4.5.3.6. Tolerancia a fallos y robustez	9
4.5.3.7. Aplicabilidad en diversas áreas	9
4.5.4. Aplicaciones con Redes Neuronales en el Procesamiento de Lenguaje Natural. 9	
4.5.5. Tipos de Redes Neuronales en el Procesamiento de Lenguaje Natural. ...	10
4.5.5.1. Redes Neuronales Recurrentes.....	10

4.5.5.2.	Redes neuronales convolucionales (CNN)	11
4.5.5.3.	Redes neuronales transformer	11
4.5.5.4.	Redes neuronales generativas(GAN)	12
4.5.5.5.	Redes neuronales Siamesas (SNNs)	13
4.5.6.	Tabla Comparativa entre los tipos de redes neuronales	14
4.6.	Metodología XP	16
4.6.1.	Pasos de la Metodología XP	16
4.6.2.	Características XP	16
Libro2.pdf		¡Error! Marcador no definido.
4.6.3.	Valores	17
Ingeniería del software-Presman 7ma edición 2010.pdf		¡Error! Marcador no definido.
4.6.4.	Proceso XP	17
4.6.4.1.	Planeación	18
4.6.4.1.1.	Historias de Usuario	18
4.6.4.1.2.	Iteraciones	18
4.6.4.1.3.	Entregas Pequeñas	19
4.6.4.1.4.	Reuniones	19
4.6.4.1.5.	Roles XP	19
4.6.4.2.	Diseño	20
4.6.4.2.1.	Tarjetas de clase, responsabilidad y colaboración (CRC)	21
4.6.4.3.	Codificación	21
4.6.4.3.1.	Programación en Parejas	21
4.6.4.4.	Pruebas	21
4.6.4.4.1.	Pruebas Unitarias	21
4.6.4.4.2.	Pruebas de Aceptación	22
4.7.	Librerías	22
4.7.1.	Sckitilearn	22
4.7.2.	Keras	22
4.7.3.	Tensorflow	23
4.7.4.	Spacy	23
4.8.	Trabajos Relacionados	¡Error! Marcador no definido.
5.	Metodología	28
5.1.	Contexto	28
5.2.	Proceso	28
5.3.	Métodos	28
5.4.	Técnicas	29
5.5.	Estándares	29

5.6. Materiales	29
6. Resultados	30
6.1. Objetivo 1:	30
Diseñar el modelo de control anti-plagio, mediante redes neuronales con Scikit-Learn.....	30
6.1.1. Iteración 1:	30
Generar el modelo de software utilizando el lenguaje Python.....	30
6.1.1.1 Fase de Análisis	30
Actividad 1: Revisión de Trabajos Relacionados.....	30
Actividad 2: Búsqueda de Librerías para trabajar con las redes neuronales.....	31
Actividad 3: Seleccionar el tipo de red Neuronal a utilizar.	32
6.1.1.2. Fase de Diseño	33
Actividad 1: Generar un diagrama con los componentes del detector de plagio. 33	
6.1.1.3. Fase de Codificación	35
Actividades: Codificar los componentes y las diferentes funciones para el análisis en documentos.	35
6.1.1.4. Fase de Pruebas	41
Actividades: Generar pruebas unitarias con textos extraídos de la web.....	41
6.1.2. Iteración 2:	43
Entrenar el modelo utilizando Scikit-Learn.	43
6.1.2.1. Fase de Análisis	43
Actividad 1: Análisis de los datos.	43
Actividad 2: Analizar el Preprocesamiento de datos.	32
6.1.2.2. Fase de Diseño	43
Actividades 1:	34
Actividades 2:	43
6.1.2.3. Fase de Codificación	43
Actividad 1: Codificar la Red Neuronal Siamesa.	44
Actividad 2: Importación de Librerías	43
Actividad 3: Carga y preparación de los Datos.....	44
Actividad 4: Creación del Modelo con Scikit-Learn.¡Error! Marcador no definido.	
6.1.2.4. Fase de Pruebas	50
Actividad 1:	50
6.1.3. Iteración 3: ¡Error! Marcador no definido.	
6.2. Objetivo 2:	53
Diseñar el software antiplagio utilizando Django y aplicando del modelo de software 4+1.....	53
6.3. Objetivo 3:	53

Evaluar la ejecución del software antiplagio, dentro de la carrera de Ingeniería en Computación, en los proyectos PIC.	53
7. Discusión	53
8. Conclusiones	¡Error! Marcador no definido.
9. Recomendaciones	53
10. Bibliografía.....	53
11. Anexos	56

Índice de Tablas
Índice de Figuras
Índice de Anexos

1. Título
2. Resumen
3. Introducción

4. Marco Teórico

- 4.1. Plagio

El plagio es una infracción de derechos de autor de cualquier trabajo producido al copiar ese trabajo sin el permiso de la persona que creó el trabajo, se puede cometer plagio de forma deliberada o de manera inconsciente. Definiciones y situaciones comunes en las cuales se comete este delito:

- El plagio ocurre cuando se toman ideas o palabras escritas por otros sin reconocer de forma directa el haberlo hecho
- Se produce al presentar como propio un trabajo de forma parcial o total sin ser el autor de dicho trabajo.
- Se considera que se comete plagio al copiar cualquier objeto de fondo, forma o incluso una simple frase.[1]

- 4.1.1. Tipos de Plagio

Hay diversos tipos de plagio y diferentes formas de describirlos. Estos se clasificarán en tres categorías y se explicarán los tipos de plagio que existen en cada una. Estas categorías se basan en la forma en que se llevan a cabo, el método utilizado y el propósito detrás de cometerlo.

- a. Tipos de plagio por su forma

- **Auto-Plagio:** Ocurre cuando un autor reutiliza sus propios escritos y los presenta como una obra inédita u original, sin citar ni referenciar sus propias publicaciones anteriores. Se destaca que simplemente agregar una referencia no es suficiente, ya que no se informa al lector ni al editor sobre el alcance de la copia. Se plantea que el término puede generar un debate ético en el ámbito académico y se relaciona con el concepto de duplicidad. El problema se enfoca más en el aspecto ético que en el legal, ya que el autor presenta una publicación como un resultado de investigación original y no informa al editor o lector sobre la cantidad o magnitud de la copia, además de no referenciar la obra previamente publicada.[2]
- **Falsa Autoría:** La falsa autoría en el contexto del plagio se refiere a la acción de atribuir erróneamente el nombre de una persona como el autor de un trabajo, como un artículo o ensayo, sin que esa persona haya realizado una contribución real en su creación. Esto implica engañar o falsificar la verdadera autoría del trabajo con el propósito de obtener reconocimiento, crédito o beneficios indebidos. Puede involucrar pagar a alguien para que escriba el trabajo y luego

presentarlo como propio, o usurpar el nombre de otra persona sin su conocimiento o consentimiento.[3]

- **Envío Doble:** es una modalidad de plagio en la que se envía y publica un mismo trabajo o resultados de investigación en varios lugares, sin reconocer ni revelar adecuadamente la existencia de la duplicidad. En términos simples, implica presentar un trabajo idéntico como si fuera original y novedoso en múltiples publicaciones, sin mencionar que ya ha sido previamente publicado o presentado en otro sitio. [4]
- **Robo de Material:** El robo de material dentro del plagio se refiere a la acción de tomar contenido de otras personas sin su autorización y sin otorgarles el crédito correspondiente. Esta forma de plagio implica apropiarse de ideas, palabras, frases, párrafos u cualquier otra forma de expresión creativa sin permiso ni reconocimiento adecuado al autor original.[5]

b. Tipos de plagio mediante métodos

- **Copiar y Pegar:** es una forma común de apropiación indebida de contenido. Consiste en copiar literalmente secciones de un texto original, ya sea en formato digital o impreso, y pegarlos en otro lugar sin realizar cambios. El copiar y pegar sin citar correctamente las fuentes o sin realizar modificaciones significativas representa una violación de los derechos de autor y de los principios éticos en el ámbito académico.[1]
- **Parafraseo inapropiado:** El plagio inapropiado se caracteriza por la ausencia de integridad académica y ética, al hacer pasar como propio lo que en realidad pertenece a otra persona. Esta conducta transgrede los fundamentos de originalidad, honestidad y respeto a la propiedad intelectual. Además, ocasiona perjuicio tanto a los autores originales como a la comunidad académica en su conjunto, al distorsionar la justa distribución del reconocimiento y los méritos por las ideas y el trabajo intelectual.[2]
- **Referencia falsa:** La referencia falsa en el contexto del plagio implica la inclusión de citas o referencias bibliográficas que no se corresponden con las fuentes reales utilizadas en un trabajo académico. Esta conducta deshonesto implica suministrar información engañosa o inventada sobre las fuentes citadas con el propósito de dar la apariencia de respaldo académico y validar las ideas expuestas.[5]

c. Tipos de plagio por su propósito

- **Intencional:** Cuando el plagio se realiza de forma intencional o premeditada, adquiere una importancia significativa, ya que implica que la persona responsable ha planeado y reflexionado sobre el acto ilícito que estaba a punto de cometer y, a pesar de ello, ha decidido llevarlo a cabo. Según McCuen, el

acto de cometer plagio no es una simple decisión que se toma de forma instantánea, sino que implica un proceso que se compone de varias etapas.[5]

- **Accidental o sin intención:** El plagio puede ocurrir de forma involuntaria cuando se olvida proporcionar la referencia de dónde se obtuvo determinada información, cuando hay confusión sobre la fuente original de información, cuando se parafrasea sin alejarse lo suficiente del texto original, lo que resulta en una similitud notable, o cuando no se realiza una referencia adecuada.[5]

4.2. Formulario de Google

4.3. Python

Python es un lenguaje de programación de propósito general, interpretado, orientado a objetos y de uso generalizado con semántica dinámica. El nombre del lenguaje de programación Python proviene de Monty Python's Flying Circus, una comedia antigua de la BBC. Guido van Rossum nació en 1956 en Haarlem, Países Bajos, y lo creó. Por supuesto, Guido van Rossum no creó ni desarrolló todas las partes de Python. Miles de programadores, testers y usuarios han trabajado arduamente para que Python se haya extendido rápidamente por todo el mundo.[6]

a. Por Qué Usar Python

Python se destaca por ser un lenguaje de programación fácil de aprender. La adquisición de habilidades en Python es significativamente más rápida que en muchos otros lenguajes.

- Python es fácil de enseñar: en comparación con otros lenguajes, la enseñanza requiere menos trabajo. Esto les permite a los maestros concentrarse más en enseñar técnicas de programación generales.[6]
- Simple de usar: se destaca por su facilidad de uso al escribir nuevo software, ya que con frecuencia es posible escribir código de forma más rápida con Python.
- Fácil de entender: se destaca por su facilidad de comprensión porque el código escrito en Python con frecuencia resulta más fácil de entender.
- Fácil de adquirir: Es un lenguaje de programación gratuito y de código abierto, lo que significa que se puede descargar y usar de forma gratuita.[6]

b. Características

- Python utiliza una sintaxis clara y fácil de leer, con un estilo de escritura que se asemeja al inglés que facilita el desarrollo y la colaboración.
- El lenguaje interpretado se ejecuta línea por línea sin compilación previa. Esto aumenta la flexibilidad y permite un ciclo de desarrollo más rápido.
- Tipado dinámico: permite cambiar el tipo de variables que se utilizan.

- Orientado a objetos: La creación de clases, la herencia de clases, la encapsulación de datos y otros conceptos de programación orientada a objetos se pueden realizar con él.
- Amplia biblioteca estándar: Python incluye una biblioteca estándar amplia y diversa que ofrece una variedad de módulos y funciones para realizar una variedad de tareas y facilita el desarrollo de aplicaciones.
- La portabilidad es amplia, lo que significa que los programas escritos en Python pueden funcionar en una variedad de sistemas operativos.
- Comunidad de desarrolladores activa: Python tiene una comunidad de desarrolladores muy activa y compasiva. La comunidad está constantemente trabajando para mejorar el lenguaje, y hay muchos recursos, bibliotecas y marcos de trabajo disponibles.[7]

4.4. Django

4.5. Redes Neuronales

El concepto de red neuronal se utiliza para describir a una serie de modelos relacionados que comparten características similares y están definidos por un amplio rango de parámetros y una estructura flexible. Estos modelos surgieron a partir de investigaciones sobre el funcionamiento del cerebro, aunque con el tiempo se han desarrollado numerosos modelos no biológicos. A pesar de esto, muchos términos utilizados en este contexto aún reflejan su origen biológico.[8]

<https://www.ibm.com/docs/es/spss-statistics/29.0.0?topic=networks-what-is-neural-network>

Una red neuronal se compone de unidades elementales PE que están interconectadas de una manera específica. El valor de las redes neuronales artificiales no solo radica en el modelo individual de cada elemento PE, sino también en la forma en que se conectan estos elementos procesadores. Por lo general, los PE se agrupan en niveles o capas, y una red típica se construye con una secuencia de capas que están conectadas entre sí de manera consecutiva.[9]

4.5.1. Modelo estándar de neurona artificial

Se va a presentar el modelo estándar de una neurona artificial basado en los principios establecidos en los trabajos de Rumelhart y McClelland (1986) y McClelland y Rumelhart (1986). Siguiendo estos principios, la neurona artificial estándar en su i-ésima forma se compone de:[10]

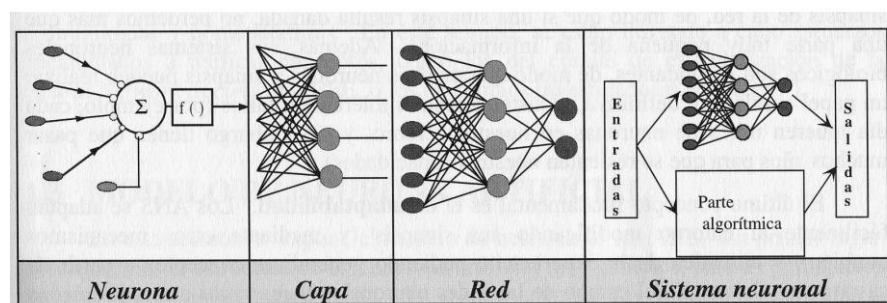


Figura 1. El proceso global en el sistema de una red neuronal.

- Se tiene un **conjunto de entradas** x_j y unos pesos sinápticos w_{ij} , donde j varía de 1 a n .
- También se cuenta con una **regla de propagación** h_i que se define utilizando dicho conjunto de entradas y pesos sinápticos. En otras palabras: $h_i(x_1, \dots, x_n, w_{i1}, \dots, w_{in})$ establece cómo se propaga la información a partir de las entradas y los pesos sinápticos.

La regla de propagación más comúnmente utilizada es combinar linealmente las entradas y los pesos sinápticos, lo cual se logra al sumar el producto de cada entrada x_i con su respectivo peso sináptico w_{ij} . De esta manera, se obtiene la siguiente expresión:

$$h_i(x_1, \dots, x_n, w_{i1}, \dots, w_{in}) = \sum (w_{ij} * x_j) \text{ para } i \text{ de } 1 \text{ a } n$$

En resumen, la regla de propagación consiste en sumar el producto de cada entrada con su respectivo peso sináptico.

Es común agregar un parámetro adicional θ_i al conjunto de pesos de la neurona, conocido como umbral. Este umbral se resta al potencial postsináptico, lo que resulta en la siguiente expresión:

$$h_i(x_1, \dots, x_n, w_{i1}, \dots, w_{in}) = \sum (w_{ij} * x_j) - \theta_i \text{ para } i \text{ de } 1 \text{ a } n$$

En resumen, el umbral se resta al potencial postsináptico generado por la combinación lineal de las entradas y los pesos sinápticos en la regla de propagación.

Si asignamos los índices i y j para comenzar en 0, y denotamos por $w_{i0} = \theta_i$ y $x_0 = -1$, podemos expresar la regla de propagación de la siguiente manera:

$$h_i(x_1, \dots, x_n, w_{i1}, \dots, w_{in}) = \sum (w_{ij} * x_j) = \sum (w_{ij} * x_j) - \theta_i$$

En resumen, al considerar los índices i y j comenzando en 0 y al introducir el parámetro θ_i como w_{i0} , y el valor x_0 como -1, podemos expresar la regla de propagación como la suma de los productos de los pesos y las entradas, restando el umbral θ_i correspondiente.

- Una **función de activación** se utiliza para representar tanto la salida de la neurona como su estado de activación. En resumen, la función de activación se aplica a la suma ponderada de las entradas y los pesos sinápticos, representada por h_i , para obtener la salida y el estado de activación de la neurona.

En la Figura 2 se presenta el modelo estándar de una neurona artificial descrito anteriormente.

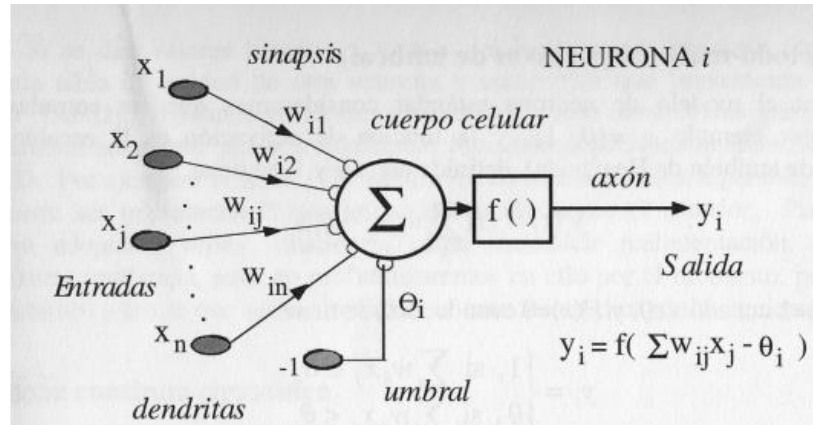


Figura 2. Modelo de neurona artificial standard.

4.5.2. Arquitectura

La arquitectura de una red neuronal se refiere a la topología, estructura o patrón de conexiones de la red. En una red neuronal artificial, los nodos se conectan mediante sinapsis, y el comportamiento de la red está determinado por la estructura de estas conexiones sinápticas. Estas conexiones son direccionales, lo que significa que la información solo puede propagarse en un solo sentido, desde la neurona presináptica a la posináptica. En general, las neuronas se agrupan en unidades estructurales llamadas capas. La red neuronal está formada por una o varias capas, y la forma en que estas capas están conectadas define la arquitectura de la red. En resumen, la arquitectura de una red neuronal se refiere a cómo están conectados los nodos mediante sinapsis, con la propagación de información en una dirección específica. Las neuronas se agrupan en capas, y el conjunto de estas capas constituye la red neuronal.[11]

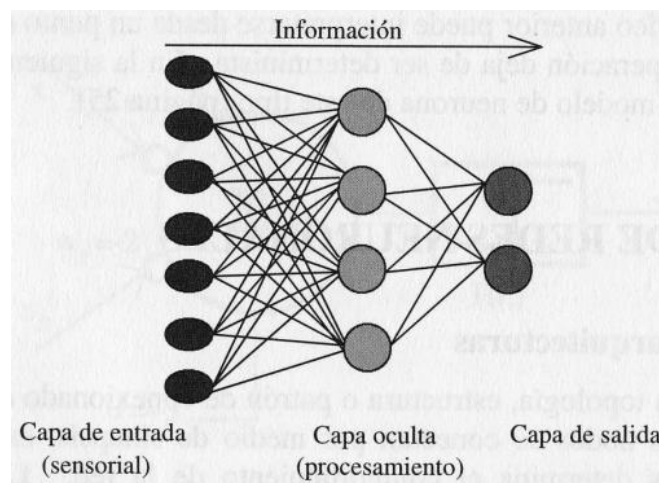


Figura 3. Arquitectura de una red Neuronal.

Capa Entrada: Esta capa es la entrada de la red neuronal y representa los datos de entrada que se utilizarán para realizar la tarea específica. Por lo general, los datos de entrada se representan como un vector numérico donde cada elemento del vector

corresponde a una característica o atributo del dato. Cada elemento del vector se conecta con los nodos de la capa de entrada.

Capa Oculta: Es el componente clave en una red neuronal, situada entre la capa de entrada y la capa de salida. En una red típica, puede haber una o varias capas ocultas, compuestas por nodos o neuronas interconectadas. Cada nodo en una capa oculta está conectado con todos los nodos de la capa anterior y la siguiente, a través de conexiones conocidas como "pesos". Estos pesos representan la fuerza o importancia de las conexiones entre los nodos. Además, cada nodo en la capa oculta tiene una función de activación que determina su salida basada en las entradas y los pesos. Estas funciones de activación pueden ser lineales o no lineales, y se utilizan para introducir no linealidades en la red neuronal. Cada capa oculta realiza operaciones de transformación en los datos de entrada mediante las conexiones y los pesos. A medida que los datos fluyen a través de las capas ocultas, la red neuronal aprende a reconocer y extraer características o patrones relevantes de los datos.[12]

Capa Salida: Esta capa es la última capa de la red neuronal y representa la salida de la red. Dependiendo del tipo de problema que se esté abordando, la capa de salida puede tener uno o varios nodos. Cada nodo en la capa de salida proporciona una estimación o predicción para una determinada variable objetivo o clase. La función de activación en la capa de salida puede variar según el tipo de problema, como una función sigmoide para problemas de clasificación binaria o una función softmax para problemas de clasificación multiclase.[13]

Función sigmoidea: La función sigmoidea da valores que oscilan entre 0 y 1. La pendiente de la función de activación se altera al cambiar el valor de la entrada g . A continuación, se tiene la fórmula de la función sigmoidea.[14]

https://www.frro.utn.edu.ar/repositorio/catedras/quimica/5_anio/orientadora1/monografas/matich-redesneuronales.pdf

$$f(x) = \frac{1}{1 + e^{-gx}}, \text{ con } x = gin_i - \Theta_i.$$

Figura 4. Función sigmoidea.

4.5.3. Ventajas de las Redes Neuronales

Las redes neuronales artificiales comparten muchas similitudes con el cerebro debido a su estructura y principios fundamentales. Por ejemplo, tienen la capacidad de aprender a partir de la experiencia, de generalizar conocimientos de casos previos a nuevos casos y de identificar características esenciales incluso en entradas que contienen información irrelevante.[14]

- **Aprendizaje Adaptativo:** Las redes neuronales son capaces de aprender y adaptarse automáticamente a partir de los datos de entrada. Pueden ajustar sus pesos y conexiones internas para mejorar su rendimiento con el tiempo.
- **Capacidad de generalización:** Las redes neuronales pueden generalizar patrones y conocimientos aprendidos de casos anteriores a nuevos casos o

situaciones similares. Esto les permite hacer predicciones y tomar decisiones en escenarios no vistos anteriormente.

- **Procesamiento paralelo:** Las redes neuronales pueden realizar múltiples cálculos simultáneamente, lo que les permite procesar grandes cantidades de información de manera rápida y eficiente. Esto es especialmente útil para tareas de procesamiento de imágenes, reconocimiento de voz y procesamiento de datos masivos.
- **Adaptabilidad y flexibilidad:** Las redes neuronales pueden adaptarse a cambios en los datos de entrada y ajustarse a nuevas condiciones. Son capaces de aprender de forma continua y actualizarse para mantener un rendimiento óptimo.
- **Manejo de datos complejos:** Las redes neuronales pueden trabajar con datos no lineales y de alta dimensionalidad, lo que las hace adecuadas para abordar problemas complejos y extraer patrones ocultos en conjuntos de datos complejos.
- **Tolerancia a fallos y robustez:** Las redes neuronales tienen la capacidad de recuperarse y mantener su rendimiento incluso si algunos de sus componentes o conexiones se dañan o se pierden. Esto las hace resistentes a fallos y más confiables en entornos ruidosos o cambiantes.

4.5.3.1. Aplicabilidad en diversas áreas

Las redes neuronales se han aplicado con éxito en una amplia gama de campos, como reconocimiento de voz, visión por computadora, procesamiento de lenguaje natural, pronóstico financiero, medicina, entre otros. Su versatilidad y capacidad de adaptación las hacen útiles en muchas disciplinas.

4.5.4. Aplicaciones con Redes Neuronales en el Procesamiento de Lenguaje Natural.

Dentro del procesamiento de textos, las redes neuronales tienen varias aplicaciones que mejoran la comprensión y el análisis del contenido textual. Algunas de estas aplicaciones son las siguientes:

- ❖ **Clasificación de texto:** Las redes neuronales se utilizan para clasificar textos en categorías específicas, como spam vs. no spam, sentimientos positivos vs. negativos, o temas específicos. Estas redes pueden aprender automáticamente patrones y características relevantes del texto para realizar una clasificación precisa.
- ❖ **Extracción de información:** Las redes neuronales pueden ser utilizadas para extraer información específica de textos, como nombres de personas, ubicaciones, fechas o eventos. Estas redes pueden aprender a identificar y etiquetar entidades y relaciones relevantes en el texto.
- ❖ **Traducción automática:** Las redes neuronales son ampliamente utilizadas en sistemas de traducción automática. Estas redes pueden aprender a traducir texto de un idioma a otro, capturando las sutilezas y las estructuras gramaticales de los diferentes idiomas para producir traducciones más precisas y naturales.

- ❖ **Generación de texto:** Las redes neuronales se utilizan para generar texto coherente y relevante en diversas aplicaciones. Pueden utilizarse para generar respuestas a preguntas, resúmenes de textos, diálogos conversacionales o incluso para la creación de contenido generado por IA.
- ❖ **Modelado de lenguaje:** Las redes neuronales se utilizan para modelar el lenguaje y predecir palabras o secuencias de palabras en un contexto dado. Estos modelos de lenguaje pueden ser utilizados para la corrección gramatical, la autocorrección de texto, la sugerencia de palabras mientras se escribe, entre otros.
- ❖ **Análisis de sentimientos:** Las redes neuronales se utilizan para analizar el sentimiento expresado en un texto. Pueden clasificar si un texto tiene un sentimiento positivo, negativo o neutro, lo que es útil en aplicaciones como la minería de opiniones, la evaluación de comentarios de usuarios, etc.
- ❖ **Reconocimiento de voz:** Las redes neuronales también se utilizan en el reconocimiento de voz, donde se convierten las señales de audio en texto escrito. Estas redes pueden aprender a reconocer patrones de habla y convertirlos en texto comprensible.[14]

4.5.5. Tipos de Redes Neuronales en el Procesamiento de Lenguaje Natural (NLP).

4.5.5.1. Redes Neuronales Recurrentes

Las Redes Neuronales Recurrentes (RNRs), también conocidas como Recurrent Neural Networks (RNN) en inglés, son modelos de redes neuronales artificiales (RNAs) en los cuales las conexiones entre las unidades están organizadas en forma de un ciclo dirigido.[15]

son especialmente adecuadas para el procesamiento de secuencias de texto, ya que pueden modelar la dependencia temporal entre las palabras. Son capaces de recordar información contextual pasada y utilizarla en el procesamiento de las palabras siguientes. Algunas variantes populares de RNN incluyen las LSTM (Long Short-Term Memory) y las GRU (Gated Recurrent Unit).[16]

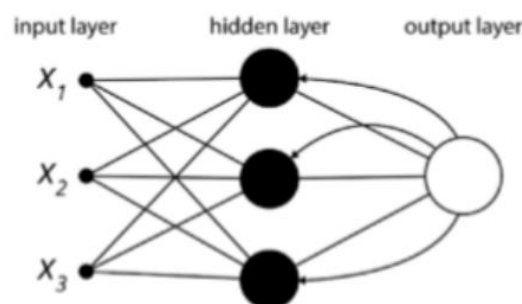


Figura 5. Arquitectura de la Red Neuronal Recurrente.

4.5.5.2. Redes neuronales convolucionales (CNN)

Las Redes Neuronales Convolucionales (CNNs), también conocidas como Convolutional Neural Networks (CNN) en inglés, son modelos ampliamente utilizados en el procesamiento de imágenes y la visión por computadora. Están diseñados de manera que imitan la estructura de la corteza visual de los animales.[17]

Aunque las CNN se asocian comúnmente con el procesamiento de imágenes, también se utilizan en el procesamiento de texto. Las capas convolucionales se utilizan para extraer características locales y patrones en secuencias de palabras, como n-gramas. Las CNN son útiles para tareas como la clasificación de texto, la extracción de información y la generación de texto.[18]

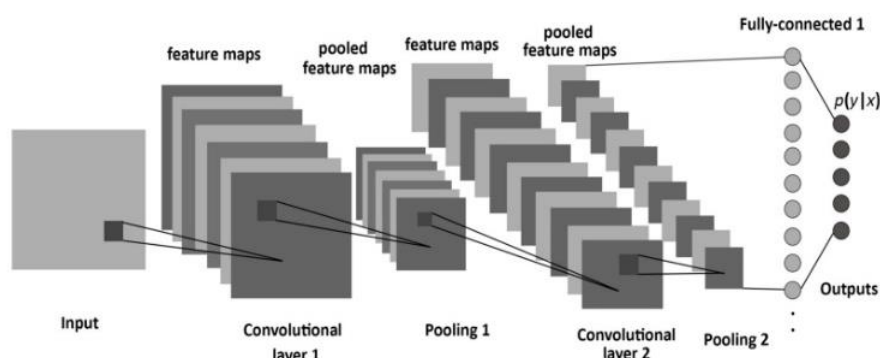


Figura 6. Arquitectura de la Red Neuronal Convolucional.

4.5.5.3. Redes neuronales transformer

Las Redes Neuronales Transformer, o simplemente Transformers, son un tipo de arquitectura de redes neuronales que se ha vuelto muy popular en el procesamiento del lenguaje natural. Fueron introducidas por primera vez en 2017 en un artículo llamado "Attention is All You Need". Los Transformers se basan en el mecanismo de atención, que permite a la red aprender y capturar relaciones entre diferentes partes de una secuencia de entrada.[19]

A diferencia de las redes neuronales recurrentes (RNN) que procesan la entrada secuencialmente, los Transformers pueden procesar todas las partes de la secuencia simultáneamente. Esto los hace más eficientes y paralelizables, lo que resulta en un procesamiento más rápido. La idea central detrás de los Transformers es utilizar mecanismos de atención para asignar pesos a las diferentes partes de la entrada, lo que permite a la red enfocarse en las partes más relevantes.

Las redes transformer han revolucionado el procesamiento de lenguaje natural. Se basan en un mecanismo de atención que permite a la red capturar relaciones de largo alcance en las secuencias de texto. Han demostrado un rendimiento destacado en tareas como la traducción automática, la generación de texto y la comprensión del lenguaje natural.[20]

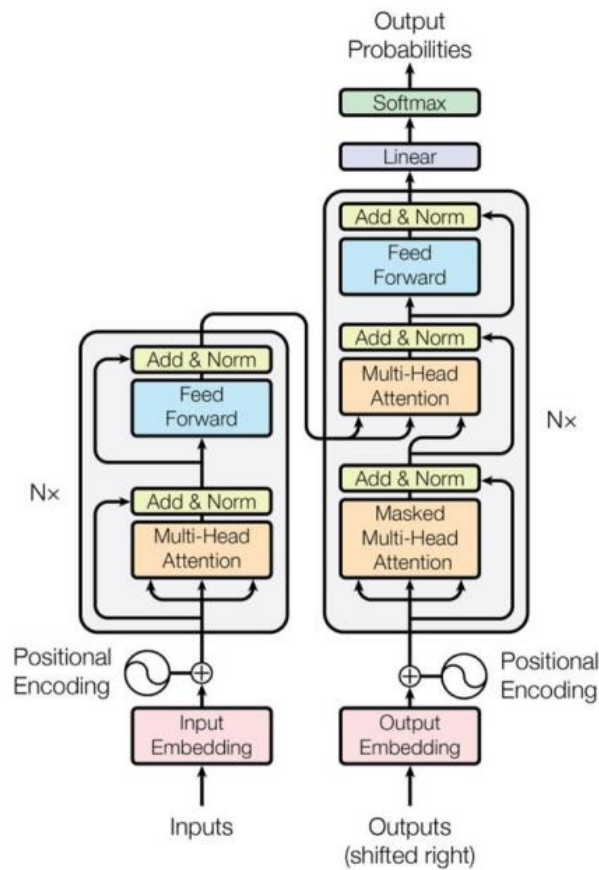


Figura 7. Arquitectura de la Red Neuronal Transformer.

4.5.5.4. Redes neuronales generativas(GAN)

Las Redes Generativas Antagónicas (GANs), conocidas como Generative Adversarial Networks en inglés, fueron presentadas en 2014 por Ian J. Goodfellow y su equipo en el artículo "Generative Adversarial Nets".[21]

Las GAN se utilizan para generar texto coherente y realista. Estas redes constan de un generador que produce muestras de texto y un discriminador que intenta distinguir entre textos reales y generados. Las GAN se aplican en la generación de texto creativo, el diálogo y la mejora de la fluidez y coherencia del lenguaje.[22]

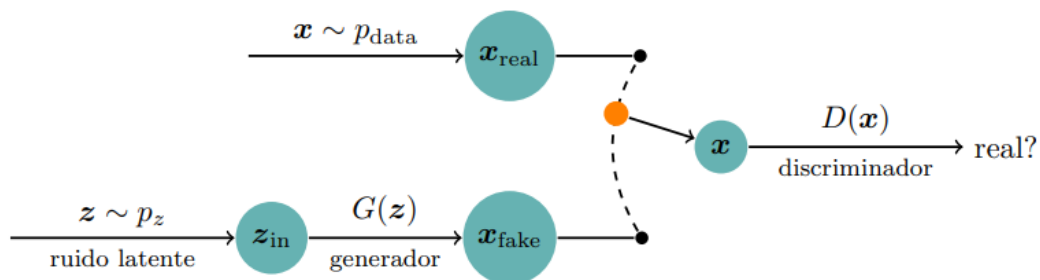


Figura 8. Arquitectura de la Red Neuronal GAN.

4.5.5.5. Redes neuronales Siamesas (SNNs)

Una red neuronal siamesa es un tipo de arquitectura de red neuronal utilizada en problemas de similitud o comparación de datos. Consiste en varias ramas o subredes idénticas que procesan las entradas de forma independiente y extraen características relevantes. Estas características se combinan en una capa de fusión para determinar la similitud entre los datos.[23]

La salida de la red siamesa se utiliza para tareas como clasificación, reconocimiento facial o detección de anomalías. Esta arquitectura permite compartir conocimientos y aprender representaciones útiles de los datos, lo que puede mejorar el rendimiento y la generalización en comparación con otras estructuras de red neuronal.[24]

Arquitectura General de la Red Neuronal Siamesa.

Sea $\{(x_i, y_i), i=1, \dots, n\}$ un conjunto de datos que consiste en n vectores de características x_i en \mathbb{R}^m de tamaño m , con etiquetas y_i en $\{1, 2, \dots, C\}$. Vamos a construir un nuevo conjunto de entrenamiento $S = \{(x_i, x_j, z_{ij})\}$ que consiste en pares de ejemplos x_i y x_j con etiquetas binarias asignadas a ellos. Si ambos vectores de características x_i y x_j son semánticamente similares, es decir, pertenecen a la misma clase, entonces z_{ij} es 0. Si los vectores son semánticamente diferentes, es decir, corresponden a diferentes clases, entonces z_{ij} es 1. Por lo tanto, el conjunto de entrenamiento S se puede dividir en dos subconjuntos: un conjunto similar o positivo con $z_{ij}=0$ y un conjunto disimilar o negativo con $z_{ij}=1$. Supongamos que las nuevas representaciones de características de los ejemplos de entrada x_i y x_j son las salidas de la red neuronal siamesa (SNN), y se denotan como h_i y h_j en \mathbb{R}^D , respectivamente. La SNN realiza un mapeo f tal que $h_i = f(x_i)$, que intenta minimizar (maximizar) la distancia euclidiana $d(h_i, h_j)$ para el par de objetos similar (disimilar) tanto como sea posible.[25] Una arquitectura estándar de la SNN se muestra en la Figura

[RS2.pdf](#)

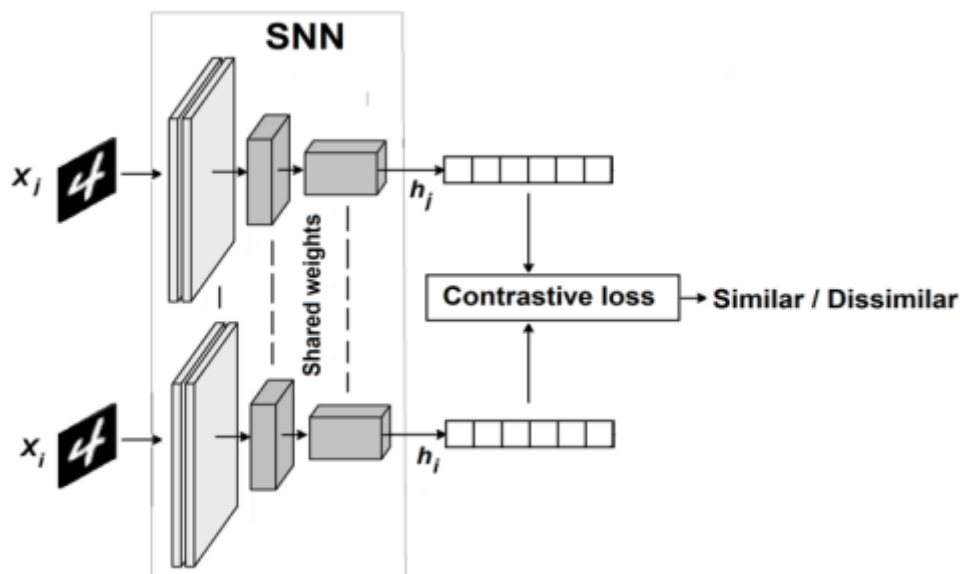


Figura 9. Arquitectura de la Red Neuronal Siamesa.

4.5.6. Tabla Comparativa entre los tipos de redes neuronales

En la siguiente tabla comparativa se resume las características principales de cada tipo de red neuronal utilizada en el procesamiento de lenguaje natural, así como sus ventajas y aplicaciones más comunes. Cada tipo de red neuronal tiene sus propias fortalezas y es adecuado para diferentes tareas dentro del campo del procesamiento de lenguaje natural.

Tabla 1. Tabla Comparativa.

Tipo de Red Neuronal	Descripción	Ventajas	Aplicaciones
Redes Neuronales Recurrentes (RNN)	Modelan dependencias temporales en secuencias de texto	<ul style="list-style-type: none"> • Capturan la información contextual y la dependencia temporal • Flexibilidad para procesar secuencias de longitud variable • Útiles en tareas de clasificación y generación de texto • Manejan información secuencial y relaciones entre palabras 	<ul style="list-style-type: none"> • Traducción automática • Generación de texto • Análisis de sentimiento • Reconocimiento de voz
Redes Neuronales	Extraen características	<ul style="list-style-type: none"> • Identifican patrones y características relevantes 	<ul style="list-style-type: none"> • Extracción de información

Convolutionales (CNN)	locales en secuencias de texto	<ul style="list-style-type: none"> • Reducción de la dimensionalidad y procesamiento eficiente • Útiles para el análisis de texto basado en n-gramas • Adecuadas para tareas de clasificación y detección de texto 	<ul style="list-style-type: none"> • Generación de texto • Resumen automático
Redes Neuronales Transformer	Modelan la relación entre palabras a través de la atención	<ul style="list-style-type: none"> • Capturan relaciones de largo alcance en secuencias de texto • Permiten una mejor comprensión del contexto global • Mayor capacidad de paralelización en el entrenamiento • Logran resultados sobresalientes en diversas tareas de NLP 	<ul style="list-style-type: none"> • Traducción automática • Generación de texto • Resumen automático • Comprensión del lenguaje natural
Red Neuronal Siamesa	Utiliza una arquitectura compartida para comparación o similitud de texto.	<ul style="list-style-type: none"> • Aprovecha el aprendizaje conjunto de pares de texto • Aprende representaciones de texto más robustas y discriminativas • Eficaz para problemas de clasificación y generación de texto • Puede manejar múltiples idiomas y dominios • Mejora la capacidad de generalización 	<ul style="list-style-type: none"> • Comparación de similitud de texto • Detección de plagio • Extracción de características • Reconocimiento de entidades nombradas • Clasificación de texto

La Red Neuronal Siamesa es una arquitectura especializada en problemas de comparación o similitud de texto. Utiliza dos o más ramas idénticas que comparten parámetros y aprenden representaciones de texto más robustas y discriminativas. Al utilizar el aprendizaje conjunto de pares de texto, puede capturar características importantes y generar representaciones de texto más eficientes.[26]

La Red Neuronal Siamesa es ampliamente utilizada en tareas como la comparación de similitud de texto, detección de plagio, extracción de características, reconocimiento de entidades nombradas y clasificación de texto. Además, su capacidad para manejar múltiples idiomas y dominios la hace muy versátil en diferentes aplicaciones del procesamiento de lenguaje natural.[27]

4.6. Metodología XP

La programación extrema, también conocida como Extreme Programming (XP), es una metodología de desarrollo de software creada por Kent Beck, quien escribió el primer libro sobre el tema, *Extreme Programming Explained: Embrace Change* (1999).[28]

XP es uno de los enfoques ágiles más reconocidos para el desarrollo de software. Al igual que otros métodos ágiles, la programación extrema se distingue de las metodologías tradicionales al enfocarse más en la adaptabilidad que en la probabilidad.[29]

4.6.1. Pasos de la Metodología XP

- **Desarrollo incremental e iterativo:** Se realizan mejoras pequeñas de forma progresiva.
- **Pruebas unitarias continuas:** Se realizan pruebas frecuentes y automatizadas, donde se incorpora las pruebas de regresión. Se recomienda escribir las pruebas antes de codificar.
- **Programación en parejas:** Se sugiere que dos personas trabajen juntas en una misma tarea de desarrollo.
- **Integración frecuente con el cliente o usuario:** Se recomienda una colaboración continua y cercana entre el equipo de desarrollo y un representante del cliente, con el fin de lograr una integración frecuente con el cliente o usuario.
- **Corrección de errores antes de agregar nuevas funcionalidades.**
- **Refactorización del código:** Se reescriben partes del código para mejorar su legibilidad y mantenibilidad sin cambiar su comportamiento. Las pruebas aseguran que no se introduzcan errores durante la refactorización.
- **Propiedad compartida del código:** Todos los miembros del equipo tienen la responsabilidad de corregir y ampliar cualquier parte del proyecto. Las pruebas de regresión garantizan la detección de posibles errores.
- **Simplicidad en el código:** Se prioriza la simplicidad para que las cosas funcionen. Si es necesario, se pueden añadir más funcionalidades posteriormente. La programación extrema (XP) se enfoca en la simplicidad y está dispuesta a realizar un esfuerzo adicional para realizar cambios si se requiere algo más complejo que posiblemente nunca se llegue a utilizar.[30]

4.6.2. Características XP

- La principal diferencia de esta metodología con respecto a las tradicionales es su mayor enfoque en la adaptabilidad en lugar de la previsibilidad.
- Se implementa de forma dinámica a lo largo del ciclo de vida del software y tiene la capacidad de adaptarse a los cambios en los requisitos.

- Se enfatiza la importancia de los individuos y las interacciones por encima de los procesos y herramientas.
- Propone un enfoque iterativo en el que se siguen cuatro pasos: planificación, diseño, codificación y prueba.
- En cada iteración, se agregan nuevas funcionalidades al software. Esta metodología es especialmente adecuada para proyectos con requisitos ambiguos y en constante cambio, así como para aquellos con un alto riesgo técnico.[29]

4.6.3. Valores

Beck (Bec04a) establece cinco valores fundamentales que son la base de todo el trabajo realizado en XP: comunicación, simplicidad, retroalimentación, valentía y respeto. Cada uno de estos valores impulsa las actividades, acciones y tareas específicas en XP.[31]

- Comunicación: La comunicación es de vital importancia en el desarrollo de software en equipo. La comunicación es esencial para crear un sentido de equipo y fomentar una cooperación efectiva. Sin embargo, aunque la comunicación es crucial, no es el único elemento necesario para lograr un desarrollo de software efectivo.
- Simplicidad: La simplicidad es el valor más enfocado en la mente y el intelecto dentro de XP. Es un desafío duro lograr que un sistema sea lo suficientemente simple como para abordar de manera elegante el problema presente. El objetivo es crear un diseño sencillo que pueda ser fácilmente implementado en forma de código. En caso de ser necesario, se realizará una reevaluación del diseño en un momento posterior.
- Retroalimentación: En un equipo ágil, la retroalimentación proviene de tres fuentes principales: el software implementado, el cliente y otros miembros del equipo de desarrollo de software. Al establecer y aplicar una estrategia de pruebas efectiva, el equipo ágil obtiene retroalimentación valiosa a través de los resultados de las pruebas realizadas en el software.[29]
- Valentía: Implica que los desarrolladores deben estar dispuestos a enfrentar el cambio, ya que es inevitable. Sin embargo, contar con una metodología establecida ayuda a manejar ese cambio de manera más efectiva.
- Respeto: El equipo ágil fomenta un ambiente de respeto entre sus miembros, así como hacia otros participantes y el propio software. A medida que el equipo logra realizar entregas exitosas de incrementos de software, se fortalece el respeto por el proceso de XP.[28]

4.6.4. Proceso XP

La programación extrema se basa en la orientación a objetos como el paradigma de desarrollo preferido, e implica un conjunto de reglas y prácticas que se aplican dentro de las actividades estructurales de planificación, diseño, codificación y pruebas.[31]

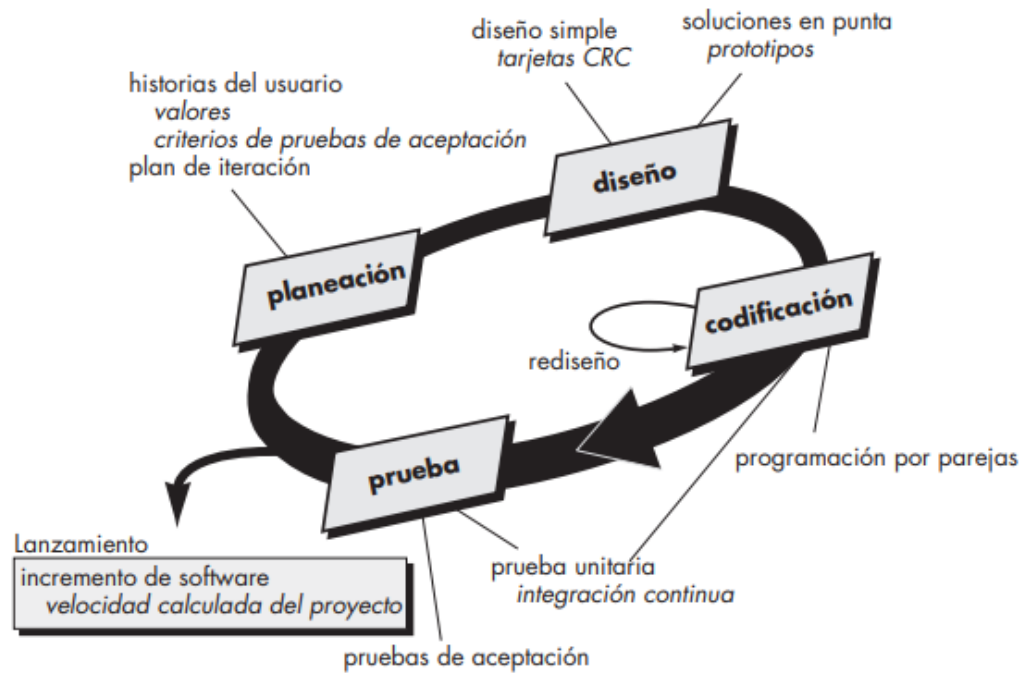


Figura 10. Proceso de la Metodología XP.

4.6.4.1. Planeación

Se inicia con una actividad de recopilación de requisitos, que tiene como objetivo comprender el contexto comercial del software y obtener una comprensión clara de las salidas deseadas, así como de las características y funcionalidades principales requeridas. Esta recopilación se lleva a cabo a través de historias de usuarios, las cuales describen los resultados, características y funcionalidades necesarias del software en desarrollo. Estas historias son redactadas por el cliente y se agregan al backlog del proyecto. El cliente asigna un valor a cada historia, basado en el valor general para el negocio o la función de la empresa, lo que determina su prioridad. A lo largo del proceso, el cliente tiene la capacidad de agregar nuevas historias, realizar modificaciones en los valores de las historias existentes, dividir las o incluso eliminarlas según sea necesario. Esto permite que el equipo de XP realice revisiones constantes de las entregas pendientes y ajuste sus planes en consecuencia.[32]

4.6.4.1.1. Historias de Usuario

4.6.4.1.2. Iteraciones

En general, los proyectos se dividen en etapas o iteraciones, lo que caracteriza a una metodología iterativa. Cada iteración tiene una duración ideal de una a tres semanas. En cada iteración, se establece un grupo de historias a implementar. Cuando se culmina la iteración, se entrega la historia correspondiente, el mismo que debería haber pasado las pruebas de aceptación establecidas por el cliente para asegurar que se cumplió con los requisitos. Las tareas pendientes que no se completaron en una iteración se tienen en consideración para la siguiente iteración. En este proceso, se colabora con el cliente para evaluar si es necesario llevar a cabo estas tareas en el futuro o si es más apropiado eliminarlas de la planificación del sistema.[33]

4.6.4.1.3. Entregas Pequeñas

El período de tiempo asignado para una iteración típicamente oscila entre una y tres semanas, al finalizar la cual se realiza una entrega del producto que debe incluir avances significativos y plenamente funcionales. Es importante destacar que estas entregas deben ocurrir con frecuencia a lo largo del desarrollo del proyecto.[33]

4.6.4.1.4. Reuniones

La planificación desempeña un papel fundamental en XP, lo que implica una revisión constante del plan de trabajo. Aunque XP promueve la reducción de documentación excesiva, se mantiene rigurosa en la organización de las tareas y actividades.

- **Plan de Entregas:** Al inicio del proyecto, se lleva a cabo una reunión conjunta entre el equipo de desarrollo y los clientes para establecer el cronograma del proyecto. Durante esta reunión, el cliente presenta las historias de usuario al equipo, quienes evalúan la complejidad de implementar cada historia.
- **Inicial de Iteración:** Al iniciar una iteración, se lleva a cabo una reunión de planificación donde se establecen las actividades de programación que serán abordadas. Durante esta reunión, las historias de usuario se desglosan en tareas más pequeñas y se asignan a los miembros del equipo de desarrollo. Cada tarea es evaluada por los desarrolladores, estimando el tiempo necesario para su finalización, usualmente en un rango de uno a tres días de programación sin interrupciones.[33]

4.6.4.1.5. Roles XP

En la metodología XP se adopta la noción de roles para organizar y asignar las diferentes actividades requeridas durante el desarrollo del proyecto. Cada uno de estos roles es desempeñado por uno o más miembros del equipo, y existe la opción de rotar los roles entre los integrantes a lo largo del ciclo de vida del sistema. Esta práctica permite una distribución eficaz de las responsabilidades y proporciona flexibilidad en la asignación de tareas dentro del equipo.

- **Programador:** En el contexto de XP, el programador desempeña el rol de crear y ejecutar pruebas unitarias, así como desarrollar el código del sistema. Su responsabilidad principal radica en garantizar que las pruebas unitarias sean efectivas y exhaustivas, y que el código producido cumpla con los requisitos establecidos. Además, el programador debe mantener una comunicación fluida y una coordinación estrecha con los demás miembros del equipo, como el cliente, el encargado de pruebas y el encargado de seguimiento, para garantizar una colaboración efectiva y un desarrollo exitoso del proyecto.[33]
- **Cliente:** el cliente desempeña un papel fundamental en la definición y validación de los requisitos del sistema. Su responsabilidad principal radica en redactar las historias de usuario, que describen las funcionalidades deseadas del sistema, y elaborar las pruebas funcionales para evaluar su implementación. Además, el cliente tiene la facultad de asignar prioridades a las historias de usuario, determinando cuáles se implementarán en cada iteración en función del valor que aporten al negocio. Es importante destacar que, aunque el cliente suele ser

una persona específica en el proyecto, puede representar a un grupo más amplio de personas que serán afectadas por el sistema.[33]

- **Encargado de pruebas (Tester):** Su función principal es colaborar estrechamente con el cliente en la elaboración de las pruebas funcionales que validarán la implementación del sistema. El encargado de pruebas realiza periódicamente estas pruebas y comparte los resultados con el equipo de desarrollo. Además, asume la responsabilidad de gestionar y mantener las herramientas de soporte necesarias para llevar a cabo las pruebas de manera eficiente y efectiva.[33]
- **Encargado de seguimiento (Tracker)** Su principal responsabilidad radica en verificar la precisión de las estimaciones realizadas y el tiempo real dedicado, comunicando los resultados obtenidos para mejorar las futuras estimaciones. Además, realiza un seguimiento exhaustivo del progreso de cada iteración, evaluando la viabilidad de los objetivos establecidos en función de las restricciones temporales y de recursos presentes. Si es necesario, el encargado de seguimiento identificará los momentos oportunos para implementar cambios y ajustes en aras de alcanzar los objetivos establecidos para cada iteración.[33]
- **Entrenador (Coach):** Su función principal es proporcionar orientación y dirección a los miembros del equipo, asegurando que se sigan las prácticas de XP de manera adecuada y que se cumplan los objetivos del proceso. El entrenador se asegura de que el equipo esté alineado con los principios y valores de XP, y trabaja en estrecha colaboración con todos los miembros para garantizar un desarrollo eficiente y exitoso del proyecto.[34]
- **Consultor:** Su rol consiste en orientar y asesorar al equipo en la resolución de problemas particulares relacionados con dicha área de conocimiento. El consultor trabaja en estrecha colaboración con el equipo, proporcionando información técnica y estratégica para superar desafíos específicos y garantizar la calidad y eficiencia del desarrollo del proyecto.[32]
- **Gestor (Big boss):** Su función principal radica en facilitar un entorno de trabajo efectivo y crear las condiciones adecuadas para el éxito del proyecto. El gestor se encarga de coordinar las actividades del equipo, asegurándose de que exista una comunicación fluida y una colaboración eficiente entre todas las partes involucradas. Su labor implica la supervisión del progreso del proyecto, la gestión de recursos y la resolución de posibles conflictos o desafíos que puedan surgir durante el proceso de desarrollo.[30]

4.6.4.2. Diseño

El diseño en XP se adhiere firmemente al principio de "mantenlo sencillo" (MS). En lugar de optar por soluciones más complejas, siempre se prefiere un diseño más simple. El diseño actúa como una guía para la implementación de una historia de acuerdo con su descripción. Si se detecta un problema de diseño complicado en una historia, XP recomienda crear un prototipo inmediato para esa parte del diseño. Esto

tiene como objetivo mitigar el riesgo antes de iniciar la implementación completa y evaluar las estimaciones iniciales de la historia que presenta el problema de diseño. XP promueve el uso de tarjetas de calificación de contribuyente responsable (CRC) como una forma efectiva de desarrollar y mejorar el software.[30]

4.6.4.2.1. Tarjetas de clase, responsabilidad y colaboración (CRC)

4.6.4.3. Codificación

Durante la fase de codificación en XP, se sigue un enfoque específico. En lugar de comenzar directamente a escribir el código, se llevan a cabo pruebas unitarias para cada historia antes de su implementación. Estas pruebas brindan retroalimentación inmediata a los desarrolladores y les permiten enfocarse en la implementación necesaria para que la historia pase la prueba. Uno de los aspectos más destacados en esta etapa es la programación en parejas, donde dos personas trabajan juntas en una estación de trabajo. Esto facilita la resolución de problemas en tiempo real y asegura la calidad del código a medida que se desarrolla. A medida que las parejas de programadores completan su trabajo, el código se integra con el trabajo realizado por otros miembros del equipo. Esto se lleva a cabo a diario mediante un equipo de integración o directamente por las parejas de programadores. La estrategia de "integración continua" ayuda a evitar problemas de compatibilidad e interfaces, y también proporciona un entorno de prueba inicial para detectar errores de manera oportuna.[31]

4.6.4.3.1. Programación en Parejas

En XP, se requiere que todo el código sea desarrollado por pares de programadores que trabajen juntos frente a un único ordenador. Aunque esto podría parecer una reducción del 50% en la productividad inicialmente, según XP, no se experimenta tal pérdida. Se sostiene que no hay una gran diferencia, en términos de cantidad, entre el código producido por una pareja de programadores en estas condiciones y el código creado por los mismos miembros trabajando de forma individual. Al trabajar en parejas, se logra un diseño de mayor calidad, un código más organizado y con menos errores que si se trabajara de manera individual. Además, se aprovecha la ventaja de tener a un compañero que puede ayudar a resolver problemas durante el proceso de codificación, los cuales surgen con frecuencia.[34]

4.6.4.4. Pruebas

La metodología XP dedica una atención significativa a las pruebas, categorizándolas en distintos tipos y asignándoles funciones específicas. Además, establece directrices precisas sobre quién debe encargarse de realizar las pruebas, cuándo deben llevarse a cabo y cómo deben ser ejecutadas.[30]

4.6.4.4.1. Pruebas Unitarias

Las pruebas unitarias son aplicadas a todos los métodos relevantes de las clases en el proyecto, asegurando que cada clase esté acompañada de su propio conjunto de pruebas. Es un requisito fundamental que ninguna clase sea liberada sin tener sus pruebas correspondientes. Un aspecto crucial de estas pruebas es que se recomienda

construirlas antes de implementar los propios métodos. Esto proporciona al programador una comprensión clara de lo que debe programar y le permite conocer todos los casos de prueba que deben superarse. Este enfoque optimiza el trabajo del programador y garantiza la calidad del código resultante.[34]

4.6.4.4.2. Pruebas de Aceptación

Las pruebas de aceptación, también denominadas pruebas funcionales, son realizadas por el cliente con base en los requisitos establecidos en las historias de usuario. En cada iteración, se seleccionan las historias de usuario que el cliente desea abordar, y para cada una de ellas se definen una o más pruebas de aceptación. Estas pruebas permiten identificar y corregir posibles errores, asegurando que el software cumpla con los requisitos y expectativas del cliente.[34]

4.7. Librerías

4.7.1. Scikit-learn

Scikit-learn es una biblioteca de Python que combina varios algoritmos avanzados de aprendizaje automático para abordar problemas supervisados y no supervisados de tamaño moderado. Su principal objetivo es poner el aprendizaje automático al alcance de personas sin conocimientos especializados empleando un lenguaje de programación de alto nivel y fácil de usar.[35]

La popularidad de scikit-learn en la investigación académica se debe en gran medida a su API bien documentada, fácil de usar y versátil. Los desarrolladores pueden realizar experimentos rápidos con diferentes algoritmos realizando pequeños cambios en el código. scikit-learn incluye implementaciones populares de algoritmos de aprendizaje automático, como LIBSVM y LIBLINEAR. Además, otras bibliotecas de Python, como NLTK, han incorporado envoltorios para scikit-learn.[36]

4.7.2. Keras

Keras es una API de alto nivel para redes neuronales, diseñada para facilitar la experimentación rápida. Ofrece una variedad de características clave:

- **Compatibilidad fluida:** Keras permite que el código se ejecute sin problemas tanto en la CPU como en la GPU.
- **Interfaz fácil de usar:** La API está diseñada para ser intuitiva, lo que facilita la creación rápida de prototipos de modelos de aprendizaje profundo.
- **Soporte integrado:** Keras proporciona soporte integrado para redes convolucionales (utilizadas en visión por computadora), redes recurrentes (para procesamiento de secuencias) y cualquier combinación de ambas.
- **Flexibilidad en las arquitecturas de red:** Keras admite diversas configuraciones de red, incluyendo modelos con múltiples entradas o salidas, compartición de capas y modelos compartidos. Esto lo hace adecuado para construir una amplia gama de modelos de aprendizaje profundo, desde redes de memoria hasta máquinas neuronales de Turing.

- **Compatibilidad con backends:** Keras puede ejecutarse en varios backends, como TensorFlow, CNTK o Theano. Esta flexibilidad permite a los usuarios elegir el backend más adecuado para sus necesidades específicas.[37]

4.7.3. Tensorflow

TensorFlow es una biblioteca de código abierto ampliamente utilizada en el campo del aprendizaje automático (machine learning) y se utiliza principalmente para desarrollar, entrenar y desplegar modelos de aprendizaje automático.[38]

Es una biblioteca matemática simbólica que permite realizar cálculos numéricos de alto rendimiento utilizando gráficos computacionales. Estos gráficos representan las operaciones matemáticas como nodos y las variables como tensores multidimensionales, de ahí el nombre "TensorFlow".[39]

4.7.4. Spacy

spaCy, desarrollado por los programadores Matthew Honnibal e Ines Montani, es una biblioteca de software de código abierto para PNL avanzada. spaCy es un marco relativamente nuevo, pero una de las bibliotecas más potentes y avanzadas que se utilizan para implementar la PLN.

Una de las características más destacadas de spaCy es que incluye modelos lingüísticos y vectores de palabras preformados para más de 60 idiomas. A diferencia de sus predecesores más académicos, como NLTK, spaCy se centra principalmente en la producción y el envío de código.

4.7.5. Otras librerías Adicionales

- **numpy:** Para realizar operaciones matemáticas y numéricas eficientes.
- **googlesearch:** Proporciona funciones para realizar búsquedas en Google.
- **bs4 (BeautifulSoup):** Librería para analizar y extraer información de documentos HTML y XML.
- **nltk:** Librería de procesamiento de lenguaje natural que contiene herramientas y recursos para el análisis de texto.

4.8. Descripción de las Librerías para trabajar con Redes Neuronales

En la tabla...se presenta una descripción detallada sobre las características principales de las librerías mencionadas, así como una justificación sobre la elección de utilizar estas librerías en particular. También se incluye información sobre los lenguajes de programación compatibles con cada una de ellas, entre otros detalles relevantes.

Tabla 2. Descripción de Librerías.

Librerías	Características	Facilidad de Uso	Modelos Pre-	Lenguaje de	Justificación

			entrenados	Programación	
spaCy	<ul style="list-style-type: none"> • Spacy es una biblioteca de alta velocidad de procesamiento de lenguaje natural de código abierto. • Tokenización, lematización, etiquetado gramatical, reconocimiento de entidades nombradas y muchas otras funciones comunes de procesamiento de texto están disponibles. • Se basa en Python y se concentra en la eficiencia y el rendimiento. • Proporciona modelos previamente entrenados para una variedad de idiomas que se pueden utilizar para realizar tareas específicas de (NLP). 	Ofrece una API intuitiva y bien documentada	Si, proporciona modelos pre-entrenados para más de 60 idiomas	Python	<ul style="list-style-type: none"> • Eficiencia y velocidad en el análisis y procesamiento de texto. • Amplia gama de modelos pre-entrenados para diferentes tareas de NLP. • Ofrece una API intuitiva y bien documentada, lo que facilita su implementación.
scikit-learn	<ul style="list-style-type: none"> • Scikit-learn es una biblioteca de aprendizaje automático basada en Python que también incorpora características para el NLP. • Proporciona herramientas para la extracción de características de texto. • Los algoritmos de aprendizaje supervisado y no supervisado se ofrecen para la clasificación y otras tareas de procesamiento de texto. • Ofrece herramientas para la selección de 	Proporciona una interfaz coherente y fácil de usar	-No	Python	<ul style="list-style-type: none"> • Amplia colección de algoritmos de aprendizaje automático para tareas de clasificación y regresión. • Proporciona una interfaz fácil de usar y coherente para construir modelos de NLP. • Documentación detallada y abundante con ejemplos prácticos y casos de uso.

	parámetros y la evaluación de modelos.				
Keras	<ul style="list-style-type: none"> • Keras es una biblioteca de aprendizaje profundo que se basa en Python y está disponible de código abierto. • Keras se puede ejecutar sobre una variedad de bibliotecas de cálculo numérico, como TensorFlow y Theano, lo que le permite aprovechar todas las características de estas bibliotecas de cálculo numérico que ya existen. • Keras admite una variedad de arquitecturas de redes neuronales que se pueden utilizar en tareas de procesamiento natural de la información (NNLP), incluidos modelos de atención, redes neuronales Siamesas, redes neuronales convolucionales (CNN) y redes neuronales recurrentes (RNN). 	Permite una fácil definición y entrenamiento de modelos de aprendizaje profundo	-No	Python	<ul style="list-style-type: none"> • Interfaz de alto nivel para construir y entrenar modelos de aprendizaje profundo. • Permite la creación rápida de arquitecturas de redes neuronales para tareas de NLP. • Amplia comunidad de usuarios y abundante documentación con ejemplos y guías paso a paso.
TensorFlow	<ul style="list-style-type: none"> • TensorFlow es una biblioteca de aprendizaje automático y profundo de código abierto. • Ofrece una plataforma completa para el desarrollo y el entrenamiento de modelos de aprendizaje automático, como el NLP. • Ofrece una variedad de API y herramientas para crear y ejecutar redes neuronales. 	Ofrece una amplia gama de herramientas y recursos para el desarrollo de modelos de aprendizaje	-No	Python, C++, Java, JavaScript, Swift, y otros	<ul style="list-style-type: none"> • Framework versátil y escalable para el desarrollo de modelos de aprendizaje automático en NLP. • Soporte para la creación de redes neuronales profundas y su entrenamiento en grandes volúmenes de datos. • Amplio ecosistema de herramientas y

	<ul style="list-style-type: none"> • Permite el desarrollo de modelos de alto y bajo nivel a través de Keras y su API de grafo computacional. • Es popular entre la comunidad de investigación y cuenta con una amplia red de bibliotecas complementarias. 	automático			recursos para el NLP.
Gensim	<ul style="list-style-type: none"> • El modelado de temas y el procesamiento de texto a gran escala son los dominios principales de Gensim. • Hace posible la implementación efectiva de algoritmos de modelado como Latent Dirichlet Allocation (LDA) y el análisis semántica oculta (LSA). • Además, Gensim incluye herramientas como Word2Vec y FastText para crear y entrenar modelos de representación de palabras. • Es una biblioteca ligera y fácil de usar centrada en la escalabilidad y el rendimiento. • La búsqueda de información, la comparación de documentos y la extracción de temas son tareas de NLP en las que Gensim es particularmente útil. 	Proporciona algoritmos eficientes para la extracción de temas y creación de modelos de word embeddings (incluye estacionamiento de palabras)	- No	Python	<ul style="list-style-type: none"> • Enfoque en el modelado de temas y la vectorización de texto. • Proporciona algoritmos eficientes para la extracción de temas y creación de modelos de word embeddings. • Documentación detallada y guías de uso para implementar técnicas de NLP específicas.

4.9. Trabajos Relacionados

4.9.1. Detección de plagio en documentos árabes: Enfoques, arquitectura y Sistemas.

El estudio de la detección del plagio ha suscitado últimamente una gran atención. Aunque los sistemas de detección de plagio pueden utilizarse con muchos idiomas diferentes, funcionan mejor cuando se crean específicamente para un idioma determinado, teniendo en cuenta sus cualidades especiales. La eficiencia de estos sistemas viene determinada en última instancia por la eficacia de sus principales módulos de procesamiento, la optimización de consultas y la reducción de documentos, que tienen un impacto significativo en los tiempos de respuesta y en la calidad de los resultados. En este ensayo, analizan varios métodos y proporcionen una arquitectura y un sistema para encontrar plagios en documentos árabes. Los cuatro niveles de los módulos principales de procesamiento de la arquitectura web propuesta permiten a los usuarios revisar los textos y analizar los resultados mediante una interfaz intuitiva.

4.9.2. Creación de un prototipo de sistema de detección de plagio con NLTK.

Este trabajo aborda el problema del plagio en entornos académicos y propone el desarrollo de un sistema de detección de plagio específico. Aunque existen métodos de detección de plagio, ninguno es óptimo para abordar la paráfrasis o la detección de la idea subyacente. Por lo tanto, proponen desarrollar un prototipo capaz de analizar textos copiados de internet y señalar los párrafos originales, incluso cuando hayan sufrido cambios superficiales. Este prototipo se estructura en módulos y utiliza conocimientos adquiridos en el área de Lenguas Aplicadas. Las tecnologías específicas utilizadas para llevar a cabo el proyecto, incluyen el lenguaje de programación Python, el corpus Wikicorpus, el entorno de desarrollo Anaconda y Spyder, así como las bibliotecas de código de Python NLTK (Natural Language Toolkit) y WordNet, junto con otras bibliotecas auxiliares. El objetivo es proporcionar una herramienta efectiva para detectar y prevenir el plagio, mejorando la integridad académica y evitando sanciones por plagio.

4.9.3. Diseño e implementación de un sistema de detección de Plagio mediante el método de similitud del coseno.

El objetivo de este proyecto de investigación es crear un sistema llamado Kipcheck que pueda calcular ecuaciones de palabras y evaluar la tasa de plagio de un documento utilizando el método del coseno. Para evitar sanciones académicas, esta evaluación tiene como objetivo garantizar que los documentos creados tengan un nivel mínimo de fraude. Kipcheck realiza tres pruebas de aplicación del sistema: cálculo máximo de palabras procesables, comparación de cálculos realizados con la aplicación y de forma manual, y verificación de la consistencia de los resultados de cálculo de la aplicación en base a esquemas diferentes.

4.9.4. Diseño e Implementación de una herramienta en Studium para evitar los plagios en asignaturas de Ingeniería.

El objetivo del proyecto es crear una solución universal y efectiva para todos los lenguajes de programación para las asignaturas de grado. Esta solución, que se basa en Studium, que permitirá al profesor activar fácilmente un sistema anticopia al crear tareas o entregas. Para reducir los gastos utilizan software libre. Los estudiantes podrán solicitar un informe sobre el análisis del sistema anticopia para ver si su tarea es similar a la de sus compañeros; sin embargo, esta función será opcional y solo estará disponible

después de enviar la tarea. El sistema de detección de plagio opera utilizando un enfoque científico conocido como "Algoritmos Locales para la Huella Digital de Documentos".

5. Metodología

5.1. Contexto

El Trabajo de Integración Curricular se llevó a cabo en la Universidad Nacional de Loja, específicamente en la Facultad de Energía, como parte del programa de la Carrera de Ingeniería en Computación. Esta iniciativa se desarrolló en colaboración con el cliente principal y director del proyecto, el señor Oscar Miguel Cumbiquis Pineda.

5.2. Proceso

El proceso para alcanzar el objetivo general del TT se detalla a continuación, mencionando cada fase de la metodología empleada para cada objetivo específico:

1. Objetivo 1: Diseñar el modelo de control anti-plagio, mediante redes Neuronales.
 - a. Fase 1: Análisis de trabajos relacionados.
 - b. Fase 2: Generar el modelo anti-plagio que se pretende desarrollar.
 - c. Fase 3: Codificar el modelo anti-plagio mediante el lenguaje Python y entrenarlo utilizando Scikit Learn.
 - d. Fase 4: Evaluar el funcionamiento del modelo mediante pruebas unitarias.
2. Objetivo 2: Diseñar el software antiplagio utilizando Django y aplicando el modelo de software 4+1.

5.3. Métodos

1. Analítico

El método analítico, consiste en descomponer un objeto de estudio en sus partes o elementos para poder analizarlos de manera individual y comprender su naturaleza, funcionamiento y relación con el todo, fue utilizado para desglosar el Trabajo de Titulación en diferentes fases, cada una de las cuales se estableció como un objetivo específico junto con sus correspondientes actividades.

2. Investigación-acción

La metodología de investigación-acción se apoya en una estructura de ciclos flexibles, ya que es importante hacer ajustes a medida que se avanza en el estudio para lograr encontrar la solución adecuada al problema. El primer paso es identificar el problema, luego se planifica una solución, se lleva a cabo su implementación y se procede a evaluar los resultados, lo que da lugar a una retroalimentación. Esta metodología se adapta a las diversas etapas del Trabajo de Titulación.

3. Métodos Observacionales

- Análisis estático: El método utilizado posibilita una revisión detallada de la estructura del producto, específicamente para el modelo de control anti plagio en la validación de la autenticidad de los TT, con el propósito de identificar posibles fallas o errores.

5.4. Técnicas

- Reuniones: Las reuniones son una técnica que implica la participación de las personas interesadas en el proyecto de titulación para revisar los progresos en cada una de las etapas, con el propósito de recibir comentarios y críticas constructivas.

5.5. Estándares

- IEEE 830: El estándar IEEE 830 consiste en un conjunto de sugerencias para la descripción de los requerimientos del software, y su uso permite cumplir con el primer objetivo específico del Trabajo de Titulación, que es definir el modelo de software para la detección de plagio en TT.

5.6. Materiales

- Recursos Humanos: Diferentes individuos provenientes de varios campos participaron en la creación del Trabajo de Titulación, lo cual se puede constatar mediante la tabla 4.

Tabla 4. Recursos Humanos.

Nombre	Descripción
Santiago Alexander Román Silva	Estudiante a cargo de la ejecución del Proyecto.
Jaime Oswaldo Paqui Medina	Estudiante a cargo de la ejecución del Proyecto.
Ing. Oscar Miguel Cumbicos Pineda	Docente director del Proyecto de Trabajo de Titulación

- Recursos de Software y Hardware: Durante la elaboración del Trabajo de Titulación se utilizaron los recursos de hardware y software detallados en la tabla 5 y 6, respectivamente.

Tabla 5. Recursos de Software.

Recurso	Descripción	Link de acceso
Google Drive	Servicio que permitió almacenar y trabajar de forma colaborativa con los diferentes documentos.	https://drive.google.com/drive/folders/1mGyMFmJINMPw4VJaY6cP1rvOzDWO3n2t?usp=sharing
Zoom	Plataforma de videoconferencia en línea que permitió participar en reuniones virtuales, trabajo colaborativo, retroalimentación del TT, etc.	https://zoom.us/
Lucidchart	Software que permitió desarrollar los diferentes diagramas.	https://www.lucidchart.com
Visual Studio Code	Software que permitió elaborar el código fuente	https://code.visualstudio.com/

Tabla 6. Recursos de Hardware.

Recurso	Descripción
Laptop	Equipo principal para el desarrollo del TT.

- Insumos: La tabla 7 presenta los elementos requeridos para llevar a cabo el Proyecto de Trabajo de Titulación.

Tabla 7. Insumos.

Insumos	Descripción
Materiales de Oficina	Los materiales de oficina se emplearon como un apoyo adicional en el proceso de elaboración del Trabajo de Titulación.
Internet	Internet se empleó para diversas actividades, tales como la comunicación, investigación y el acceso a diferentes herramientas.

6. Resultados

En este apartado se proporcionan los resultados alcanzados durante la ejecución del Trabajo de Integración Curricular. Para lograrlo, se establecieron tres objetivos, donde cada uno tiene sus correspondientes iteraciones con su respectiva actividad, las cuales se completaron en base al enfoque de la Metodología XP.

6.1. Objetivo 1:

Diseñar el modelo de control anti-plagio, mediante redes neuronales con Scikit-Learn.

6.1.1. Iteración 1:

Generar el modelo de software utilizando el lenguaje Python.

6.1.1.1 Fase de Análisis

Actividad 1: Revisión de Trabajos Relacionados.

En esta actividad se llevó a cabo un análisis de trabajos previamente desarrollados que estén directamente relacionados con el tema de estudio. El propósito de esta actividad fue obtener información pertinente y relevante del área de estudio.

Subactividades

- Identificar fuentes: En el marco de esta subactividad, se llevó a cabo un proceso sistemático de búsqueda y recopilación de fuentes bibliográficas relevantes, que abarcó una variedad de recursos como artículos científicos, tesis, informes

técnicos y libros. El objetivo fue obtener información y referencias de calidad, respaldadas por la investigación académica y científica en el campo de estudio correspondiente. Para visualizar los documentos (véase el Anexo 1).

Tabla 8. Fuentes Bibliográficas.

Fuente	Nro Documentos
IEEE Xplore	12
ACM	6
ScienceDirect	1
Google Scholar	5
Otros	10

- **Selección de Trabajos relacionados:** En esta subactividad, se llevó a cabo una meticulosa evaluación y selección de los trabajos relacionados más pertinentes y relevantes al tema de estudio. Este proceso implicó el análisis crítico de una amplia gama de fuentes académicas, científicas, artículos, investigaciones previas, y otros documentos pertinentes. La finalidad de esta evaluación y selección fue identificar y utilizar los trabajos que aportaran mayor valor, rigor metodológico y aportes significativos al desarrollo de la investigación del tema de estudio. Para visualizar los documentos (véase el Anexo 2).

Tabla 9. Selección de Trabajos Relacionados.

Fuente	Nro Trabajos Seleccionados
IEEE Xplore	1
ACM	1
ScienceDirect	0
Google sCHOOLAR	5
Otros	1

- **Lectura y Comprensión:** En esta subactividad, se llevó a cabo una lectura y análisis minucioso de los trabajos relacionados seleccionados, con el propósito de obtener un entendimiento más profundo de sus objetivos, metodología, resultados y demás aspectos relevantes. Este proceso permitió adquirir un conocimiento detallado de los trabajos relacionados.

Actividad 2: Búsqueda de Librerías para trabajar con las redes neuronales.

Se investigaron y exploraron diversas librerías y frameworks disponibles que fueran adecuados para el desarrollo y la implementación de redes neuronales en nuestro proyecto. Esta actividad se realizó con el objetivo de encontrar las herramientas más adecuadas y eficientes para trabajar con redes neuronales.

Subactividades

- **Investigación y exploración:** Se realizó una investigación exhaustiva sobre diversas librerías y frameworks que se consideraban de fácil utilización para el desarrollo y manipulación de redes neuronales. Se llevó a cabo un análisis detallado de las características, funcionalidades y documentación disponible de cada una de estas herramientas. El objetivo de esta investigación fue identificar aquellas librerías y frameworks que ofrecieran una amplia gama de funcionalidades orientadas al trabajo con redes neuronales. Para visualizar los documentos (véase el Anexo 3).
- **Análisis de características:** Se procedió a realizar un análisis de las características y capacidades de cada una de las librerías consideradas, centrándonos en aspectos relevantes como su flexibilidad para la construcción de diversos tipos de redes neuronales (como redes convolucionales, recurrentes, siamesas, etc.). Así mismo, se evaluó la compatibilidad de dichas librerías con hardware especializado, destacando especialmente su capacidad para aprovechar el rendimiento y las capacidades de aceleración proporcionadas por tarjetas gráficas u otros dispositivos de cómputo específicos. (mar teorico)
- **Selección de librerías:** Después de llevar a cabo un minucioso proceso de evaluación y comparación, se procedió a la selección de las librerías que mejor se ajustaban al requerimiento fundamental del proyecto (tensorflow,keras, scikitlearn,space). Esta elección se basó en criterios técnicos y específicos, considerando factores como la compatibilidad con el lenguaje de programación utilizado, la disponibilidad de funcionalidades requeridas, la documentación y soporte ofrecidos, así como la reputación y trayectoria de la librería en el ámbito del desarrollo de redes neuronales.

Actividad 2: Analizar el Preprocesamiento de datos.

En esta actividad se realizó un análisis para saber cómo se procesarán los datos, aplicando la tokenización, normalización y lematización. Para la parte de tokenización se pretende descomponer los textos en partes más pequeñas como por ejemplo frases. La normalización para estandarizar el texto y eliminar variaciones irrelevantes. En el contexto de detección de plagio, la normalización será de gran utilidad para asegurar de que los textos sean comparables de manera precisa. Esta técnica nos servirá para convertir todas las letras a minúsculas, eliminando así las diferencias entre mayúsculas y minúsculas, excluidos caracteres especiales, tildes. Esto permitirá evitar que se pasen por alto similitudes debido a la diferencia en la capitalización de palabras. Por último, la lematización para convertir las palabras a su forma canónica, es decir, a su forma fundamental y reconocible en el lenguaje. Por ejemplo, las palabras "corriendo", "corre" y "correrá" se lematizarían a la forma base "correr". Esto permitirá tratar todas estas variantes como una sola palabra durante la comparación de textos.

Actividad 3: Seleccionar el tipo de red Neuronal a utilizar.

En el marco de esta actividad, se procedió a realizar un análisis, investigación exhaustiva y una evaluación meticulosa de los diversos tipos de redes neuronales

existentes. El objetivo principal fue determinar cuál de estos tipos era el más adecuado y pertinente para su implementación dentro del proyecto. Este proceso implicó examinar detalladamente las características, propiedades y capacidades de cada tipo de red neuronal, así como considerar sus ventajas y limitaciones en relación con los requisitos y objetivos del proyecto.

Subactividades

- **Análisis del problema:** Se llevó a cabo un análisis detallado del problema que se buscaba resolver mediante el uso de la red neuronal. Este análisis se enfocó en comprender en profundidad la naturaleza y las características del problema, así como en identificar sus desafíos y requerimientos específicos. El objetivo de este análisis fue proporcionar una base sólida para diseñar y desarrollar la red neuronal de manera eficiente y efectiva, asegurando que estuviera adecuadamente adaptada para abordar y solucionar el problema en cuestión.
- **Investigar los tipos de redes neuronales:** Se realizó una investigación y análisis de los diversos tipos de redes neuronales que resultaban relevantes para su aplicación en el proyecto propuesto. Esta investigación implicó explorar las características, estructuras y aplicaciones de cada tipo de red neuronal. El análisis se centró en la comprensión detallada de las ventajas de cada tipo de red neuronal, con el objetivo de seleccionar una opción que se caracterizara por su facilidad de implementación. Se consideraron cuidadosamente las características y atributos que contribuyen a una implementación más eficiente y práctica, tales como la simplicidad del diseño, la disponibilidad de recursos de aprendizaje y la accesibilidad de las herramientas y bibliotecas relacionadas.
- **Seleccionar la red neuronal:** En base al análisis y la evaluación realizada, se seleccionó el tipo de red neuronal que mejor se ajustaba a las necesidades y objetivos del proyecto propuesto (Red Neuronal Siamesa). El proceso de selección se realizó de manera rigurosa, garantizando que la elección final fuera la más apropiada y adecuada para lograr los resultados buscados.

6.1.1.2. Fase de Diseño

Actividad 1: Generar un diagrama con los componentes del detector de plagio.

La actividad consistió en generar un diagrama que representara los componentes que formarían parte del detector de plagio. Se elaboró un esquema visual que muestre de manera clara y organizada los diferentes elementos que contiene el sistema de detección de plagio. El diagrama consta de los componentes esenciales del detector de plagio. Cada componente representado tiene las conexiones y relaciones necesarias para su correcto funcionamiento. El objetivo principal del diagrama es proporcionar una representación visual clara y precisa de la arquitectura y los componentes del detector de plagio, brindando una guía visual para el proceso de implementación. Durante la fase de codificación, se documentará en detalle cada uno de los componentes y se definirán las funcionalidades específicas que cada uno de ellos desempeñará en el sistema de detección de plagio.

Componentes Generales.

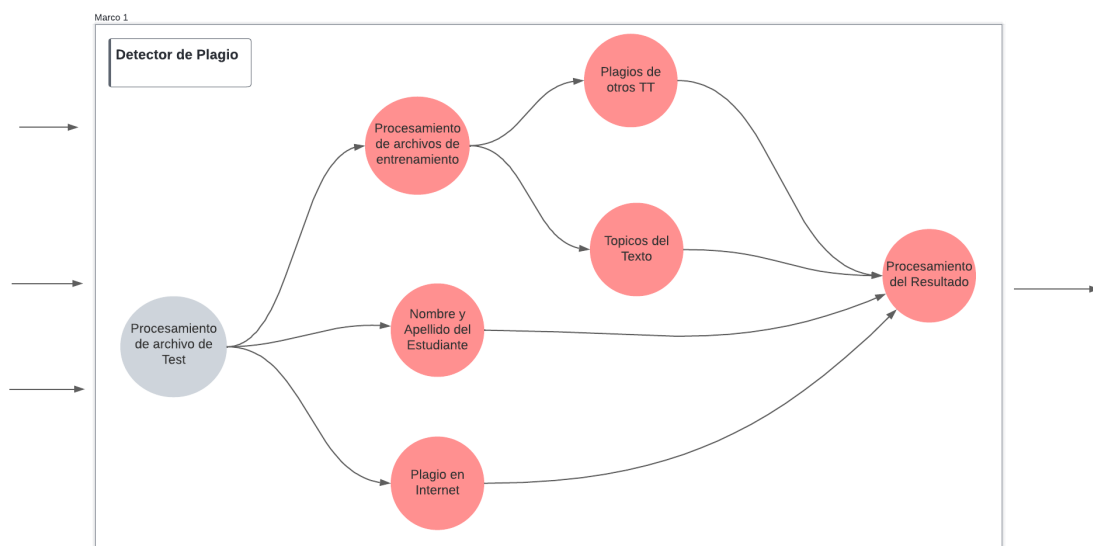


Figura 11. Componentes Generales del detector de plagio.

Actividades 1: Diseñar la Arquitectura de la Red Neuronal Siamesa

Tras realizar un análisis de los diversos tipos de redes neuronales, se optó por utilizar la red neuronal siamesa debido a su capacidad para encontrar similitudes entre archivos y su funcionamiento general (véase el marco teórico **la sección 4.5.6. Tabla Comparativa entre los tipos de redes neuronales**). Con base en estas consideraciones, se decidió trabajar con esta arquitectura para el diseño posterior (vease la imagen...), con el objetivo de crear un modelo capaz de determinar similitudes en textos. Esta arquitectura se compone de tres fases principales: la fase de entrada (Input), la fase de procesamiento (Processing) y la fase de salida (Output).

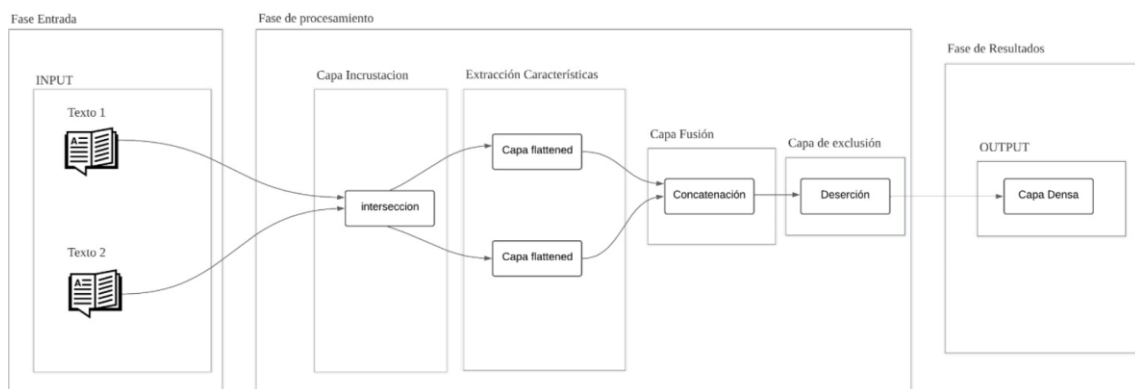


Figura 12. Arquitectura de la Red Neuronal Siamesa.

- **Fase de entrada:** Las capas de entrada Input1 e Input2 introducen los datos en la red. Representa la fase inicial, cuando los datos son recibidos y procesados por las capas subsiguientes.
- **Fase de procesamiento:** En esta fase, los datos de entrada pasan por varias capas, se transforman y se manipulan para aprender representaciones significativas. La capa de incrustación, las capas de aplanamiento, la capa de concatenación, la capa de deserción y las capas densas, son algunas de las capas involucradas en esta fase. Para hacer predicciones o capturar medidas de documentos plagiados o no plagiados, estas capas procesan colectivamente los datos de entrada, extraen características pertinentes y realizan cálculos.
- **Fase de resultados:** Es la conclusión del modelo de red neuronal, la produce la última capa densa, que utiliza las representaciones aprendidas en la fase de procesamiento para producir la salida deseada. La fase de resultados proporcionaría las probabilidades de clase de predicciones o predicciones binarias en un problema de clasificación binaria.

6.1.1.3. Fase de Codificación

Actividades: Codificar los componentes y las diferentes funciones para el análisis en documentos.

En esta actividad se procedió a la codificación de las funciones principales que implicaban la lectura de los archivos a comparar, la limpieza de los archivos, la conversión de los documentos ingresados a un formato adecuado para el procesamiento de código, el análisis de los documentos mediante técnicas de vectorización y el cálculo de la similitud del coseno para obtener el porcentaje de similitud entre ellos.

1. Campos de los archivos

- **path_archivos_referencia:** ubicación de los archivos o textos de prueba donde se desee consultar el plagio.
- **path_archivo_test:** ubicación del archivo del que queremos detectar el plagio.
- **path_resultado:** ubicación donde queremos almacenar el archivo resultado del algoritmo de detección de plagio.
- **cantidad_de_topicos:** cantidad de palabras clave, tema, o tópicos del texto de test que deseamos que el algoritmo nos diga para luego poder identificar el tema más fácilmente.
- **cantidad_de_links:** cantidad de links de Google donde queremos que el detector busque si hay plagio por oración.
- **buscar_en_pdfs:** True o False. Si queremos que, en caso de que uno de los links donde se busca plagio sea un pdf que haya que descargar, se busque plagio ahí también o no.

2. Descripción de los Componentes

Procesamiento de archivos

El algoritmo comienza obteniendo todos los archivos de entrenamiento y el archivo de test y procesandolos para su posterior manejo.

Se levantan todos los archivos de su correspondiente directorio y, los tipos de archivos soportados se parsean(análisis de sintaxis) a texto y se guardan en una clase ArchivoTxt, que funciona como una Struct de C, es decir, solamente es para almacenar cómodamente los datos para ser utilizados, en este caso un Struct de Nombre (nombre del archivo), Extensión, y Texto. Los tipos de archivos no soportados se descartan.

Luego de levantarlos se limpian todos los archivos y se obtiene de cada uno de ellos un array de oraciones, que será mucho mejor para que los procesen los siguientes componentes. Dividir el texto en oraciones se hace con la función `sent_tokenize` de `nltk` al que se le puede especificar el lenguaje en el que está el texto para que se tengan en cuenta además de puntos, signos de pregunta, exclamación y demás a la hora de dividir.

La limpieza de los archivos está separada en hilos (uno por archivo) para que se realice más rápido. Esta limpieza esta realizada con expresiones regulares e implica:

- Convertir múltiples enters (`\n+`) en un entero solo.
- Convertir un enter (`\n`) en un punto (para que `sent_tokenize` tome a los enteros como puntos).
- Convertir múltiples espacios a un espacio solo.
- Letras con tilde a su correspondiente letra sin tilde.
- Normalizar todo a unicode para que no queden caracteres extraños que puedan haber aparecido al momento del parseo, como emoticones, o imágenes. Esto se hizo con la librería `unicodedata2`.

Una vez que realizamos la limpieza de los archivos, obtenemos del archivo "Texto excluido de plagio.txt" que tiene como dijimos antes, texto que no nos gustaría que se analizara el plagio.

```

def limpieza(archivo):
    archivo_limpio = re.sub(r'\n+', '\n', archivo.strip())
    archivo_limpio = re.sub('\n', ' ', archivo_limpio.strip())
    archivo_limpio = re.sub(r'[.][.]+', '.', archivo_limpio.strip())
    archivo_limpio = re.sub(r'[ ]+[ ]+', ' ', archivo_limpio.strip())
    archivo_limpio = re.sub('á', 'a', archivo_limpio.strip())
    archivo_limpio = re.sub('é', 'e', archivo_limpio.strip())
    archivo_limpio = re.sub('í', 'i', archivo_limpio.strip())
    archivo_limpio = re.sub('ó', 'o', archivo_limpio.strip())
    archivo_limpio = re.sub('ú', 'u', archivo_limpio.strip())
    archivo_limpio = re.sub('»', '»', archivo_limpio.strip())
    archivo_limpio = re.sub('«', '«', archivo_limpio.strip())
    archivo_limpio = re.sub('\u200b', ' ', archivo_limpio.strip())

    archivo_limpio = unicodedata.normalize("NFKD", archivo_limpio.strip())

    oraciones = sent_tokenize(archivo_limpio.strip(), "spanish")
    oraciones_limpias = []
    for oracion in oraciones:
        if oracion.strip() != '.':
            if oracion.strip().endswith('.'):
                oracion_a_agregar = oracion[:-1]
            else:
                oracion_a_agregar = oracion
            oraciones_limpias.append(oracion_a_agregar.strip())

    i=0
    j=0

    oraciones_mas_limpias = []
    while i < len(oraciones_limpias):
        if i == 0:
            oraciones_mas_limpias.append(oraciones_limpias[0])
        else:
            palabras_oracion = word_tokenize(oraciones_limpias[i])
            if palabras_oracion[0].islower():
                oraciones_mas_limpias[j] += " " + oraciones_limpias[i]
            else:
                j += 1
                oraciones_mas_limpias.append(oraciones_limpias[i])
            i += 1
    return oraciones_mas_limpias

```

Figura 13. Función para la Limpieza de Archivos.

Tópicos del Texto

Para este componente esperamos a que terminen todos los hilos de procesamiento de archivos para poder arrancarlo, ya que los necesitaremos. Para obtener el tema o tópicos del texto se utilizó Topic Modelling, que se basa en obtener un modelo estadístico de los tópicos de un texto, también llamadas palabras clave, temas principales, etc. Se eligió utilizar el modelo LDA para realizar el Topic Modelling sobre el texto a analizar. El modelo de Asignación Latente de Dirichlet (LDA) es un modelo de obtención de tópicos el cual se entrena con ciertos archivos. El modelo ve a cada archivo como una serie de categorías o tópicos distribuidos en base a la distribución de Dirichlet (familia

de distribuciones de probabilidad continuas multivariada según Wikipedia). Este modelo está fuertemente relacionado con Bag of Words, ya que no le importan el orden en el que estén las palabras sino la cantidad de las mismas en el texto.

```
def obtener_tema_del_texto(archivo_test, sw, cantidad_de_topicos):
    log.info("TOPICOS | Obteniendo topicos del texto ...")
    log.debug("TOPICOS | Preparando archivos para modelo LDA ...")
    nlp = spacy.load('es_core_news_sm')

    hilos_preparar_archivos_para_lda = list()
    for archivo in archivos_referencia_limpios:
        hilo_preparar_archivo_para_lda = threading.Thread(target=preparar_archivo_referencia_para_lda,
                                                         args=(archivo, nlp, sw,))
        hilos_preparar_archivos_para_lda.append(hilo_preparar_archivo_para_lda)
        hilo_preparar_archivo_para_lda.start()

    texto_preparado_test = preparar_texto_para_lda(archivo_test, nlp, sw)

    for index, thread in enumerate(hilos_preparar_archivos_para_lda):
        thread.join()

    log.debug("TOPICOS | Archivos de referencia preparados")
    log.debug("TOPICOS | Corriendo algoritmo LDA para archivos de referencia ...")
    diccionario = corpora.Dictionary(textos_preparados_referencia)
    corpus = [diccionario.doc2bow(texto) for texto in textos_preparados_referencia]
    modelo_lda = gensim.models.LdaMulticore(corpus, num_topics=10, id2word=diccionario, passes=2)

    log.debug("TOPICOS | Algoritmo LDA para archivos de referencia finalizado")

    log.debug("TOPICOS | Corriendo algoritmo LDA para archivos de test ...")
    indice, score = sorted(modelo_lda[diccionario.doc2bow(texto_preparado_test)], key=lambda tup: -1 * tup[1])[0]

    topico_con_mas_score.extend(
        [palabra.split("**")[1].replace("\\", "") for palabra in modelo_lda.print_topic(indice, cantidad_de_topicos).split(" + ")]])

    log.info(f"TOPICOS | Topicos del texto: {topico_con_mas_score}")
```

Figura 14. Función para los Tópicos del Texto.

Obtención de los tópicos del texto.

Básicamente lo primero que hacemos es preparar cada archivo de referencia para entrenar el modelo LDA. A cada archivo se prepara en un hilo aparte para tener paralelismo de tareas. “Prepararlo para LDA” consiste en quedarse solamente con los sustantivos del texto y lematizados, es decir, con su palabra raíz (si la palabra es jirafas, lematizada sería jirafa). Esto se lo realiza con spacy, con el modelo de es_core_new_sm. Luego con la librería Gensim, se crea un diccionario que contiene la cantidad de veces que aparece una palabra en los archivos de entrenamiento y con Gensim doc2bow se crea una lista que contiene la cantidad de palabras de cada archivo y el número de veces aparecen. Ahora ya se tiene listo todo lo que se necesita para entrenar el modelo LDA, por lo que se lo entrena. Luego de entrenarlo se le envía un archivo que desconoce el modelo (el archivo del cual queremos saber sus tópicos) y se aplica al modelo entrenado. Esto me va a retornar varias listas de posibles tópicos junto con un puntaje asociado, por lo que nos quedamos con la lista de tópicos que más puntaje tenga, y con los primeros n tópicos, siendo n la cantidad de tópicos que definimos en el archivo de configuración.


```
def preparar_texto_para_lda(archivo, nlp, sw):
    archivo_mas_limpio = []
    for oracion in archivo:
        sustantivos = [token.lemma for token in nlp(oracion.lower()) if token.pos_ in ['NOUN'] and len(token.text) > 2]
        oracion_mas_limpia = [palabra for palabra in sustantivos if not palabra in sw and not str(palabra).isnumeric()]
        archivo_mas_limpio = archivo_mas_limpio + [palabra for palabra in oracion_mas_limpia if str(palabra) != '']
    return archivo_mas_limpio

def preparar_archivo_referencia_para_lda(archivo, nlp, sw):
    archivo_preparado = preparar_texto_para_lda(archivo.texto, nlp, sw)
    textos_referencia.append(ArchivoTxt(archivo.nombre, archivo.extension, archivo_preparado))
    textos_preparados_referencia.append(archivo_preparado)
```

Figura 15. Función para entrenar el modelo LDA.

Plagio de Otros TT

Tanto para este componente como el de Plagio de Internet, para la parte de comparar una oración con otra para saber si es plagio o no, se utilizó el método de Similitud Del Coseno, en el cual se obtienen resultados suficientemente favorables para lo que se requiere lograr. La similitud del coseno es una medida de la similitud existente entre dos vectores en un espacio que posee un producto interior con el que se evalúa el valor del coseno del ángulo comprendido entre ellos. Para obtener el plagio de otros trabajos prácticos, lo que hacemos es dado nuestro archivo del cual queremos detectar plagios ya previamente separado en oraciones, preparamos cada oración para ser comparada y la comparamos con cada oración (preparada para comparar) de cada archivo de entrenamiento (los otros TT).

```
def obtener_similitud_del_coseno(vector1, vector2):
    interseccion = set(vector1.keys()) & set(vector2.keys())
    numerador = sum([vector1[x] * vector2[x] for x in interseccion])

    sum1 = sum([vector1[x] ** 2 for x in vector1.keys()])
    sum2 = sum([vector2[x] ** 2 for x in vector2.keys()])
    denominador = math.sqrt(sum1) * math.sqrt(sum2)

    if not denominador:
        return 0.0
    else:
        return float(numerador) / denominador

def string_a_vector(texto):
    palabras = re.compile(r'\w+').findall(texto)
    return Counter(palabras)

def obtener_similitud(texto1, texto2):
    vector1 = string_a_vector(texto1)
    vector2 = string_a_vector(texto2)
    return obtener_similitud_del_coseno(vector1, vector2)
```

Figura 16. Función para Obtener la similitud del Coseno.

Plagio de Internet

Lo que realiza este componente es bastante similar al anterior componente, salvo que no depende de que el componente del procesamiento de los archivos

de entrenamiento haya terminado ya que no los necesita, por lo que comenzará a ejecutarse una vez que el archivo de test esté procesado. Básicamente lo que hacemos es dado nuestro archivo del cual queremos detectar plagios ya previamente separado en oraciones, preparamos cada oración para ser comparada y la buscamos en Google con la API de googlesearch. Entonces la oración ya está preparada para ser comparada como se mencionó en el componente anterior. Luego, de cada oración, nos quedamos con los primeros n links que resultaron de buscarla en Google, siendo n el número que definimos en el archivo de configuración para el campo `cantidad_de_links`. Guardamos el html de cada una de esas páginas y las convertimos a texto con la herramienta de BeautifulSoup. De cada oración se va obteniendo el porcentaje de similitud según el algoritmo mencionado anteriormente y se va guardando en una lista como `(oracion, oracion_mas_parecida, mayor_porcentaje, url_donde_se_encontro, ubicacion_dentro_de_la_pagina)`.

```
def obtener_oracion_mas_parecida_de_internet(oracion, oracion_preparada, sw, cantidad_de_links, buscar_en_pdf):
    mayor_porcentaje = 0.0
    oracion_mas_parecida = ''
    url_donde_se_encontro = ''
    archivo_donde_se_encontro = ''
    ubicacion_dentro_de_la_lista = 0

    mutex.acquire()
    for url in search(oracion_preparada, tld="com.ar", num=cantidad_de_links, stop=cantidad_de_links, pause=2):
        time.sleep(0.002)
        mutex.release()
        if (not buscar_en_pdf) and url.endswith(".pdf") or str(url).endswith(".pdf/"):
            mutex.acquire()
            continue
        else:
            texto = obtener_html_como_texto(url)
            if texto != '':
                archivo = limpieza(texto)
                for oracion_a_comparar in archivo:
                    oracion_a_comparar_preparada = preparar_oracion(oracion_a_comparar, sw)
                    if oracion_a_comparar_preparada is None:
                        continue
                    similitud = obtener_similitud(oracion_preparada, oracion_a_comparar_preparada)
                    if similitud > 0.8:
                        mayor_porcentaje = similitud
                        oracion_mas_parecida = oracion_a_comparar
                        url_donde_se_encontro = url
                        archivo_donde_se_encontro = archivo
                        ubicacion_dentro_de_la_lista = int(archivo.index(oracion_a_comparar))
                        break
                    elif similitud > mayor_porcentaje:
                        mayor_porcentaje = similitud
                        oracion_mas_parecida = oracion_a_comparar
                        url_donde_se_encontro = url
                        archivo_donde_se_encontro = archivo
                        ubicacion_dentro_de_la_lista = int(archivo.index(oracion_a_comparar))
            if mayor_porcentaje > 0.8:
                mutex.acquire()
                break
            mutex.release()
        time.sleep(0.002)
    mutex.release()
    ubicacion_principio = sum(map(len, map(word_tokenize, archivo_donde_se_encontro[:ubicacion_dentro_de_la_lista])))
    ubicacion_fin = ubicacion_principio + len(word_tokenize(oracion_mas_parecida))
    ubicacion_donde_se_encontro = f"({ubicacion_principio}, {ubicacion_fin})"
    porcentajes_de_aparicion_internet.append((oracion, oracion_mas_parecida, mayor_porcentaje, url_donde_se_encontro, ubicacion_donde_se_encontro))
```

Figura 17. Función para Obtener plagio de Internet.

6.1.1.4. Fase de Pruebas

Actividades: Generar pruebas unitarias con textos extraídos de la web.

Durante esta actividad, se llevaron a cabo pruebas unitarias con el objetivo de verificar y validar el funcionamiento preciso de las funciones implementadas en la fase de codificación. Estas pruebas se realizaron a nivel de unidades de código individual, evaluando exhaustivamente cada función para asegurar su conformidad con los requisitos establecidos y su coherencia con el diseño del sistema.

Tabla 10. Casos de Prueba para la Iteración 1.

Caso de prueba	Descripción	Texto 1	Texto 2	Resultado esperado																		
Caso 1	Plagio completo: ambos textos son idénticos	"Prueba 1"	"Prueba 2"	<p>Tópicos del texto: <i>fruta, verdura, riesgo, salud, enfermedad</i></p> <p>Análisis de plagio</p> <p>Total de 21 plagios encontrados</p> <p>Porcentaje de plagio general: 100%</p> <table><tr><th>Oración plagiada</th><th>Oración original</th><th>Lugar donde se encontró</th></tr><tr><td>Las frutas y verduras son una parte esencial de una dieta sana</td><td>Las frutas y verduras son una parte importante de una dieta sana</td><td>Prueba 2.pdf</td></tr><tr><td>Estan repletas de vitaminas, minerales y antioxidantes que pueden ayudarle a proteger su cuerpo de las enfermedades</td><td>Estan repletas de vitaminas, minerales y antioxidantes que pueden ayudarle a proteger su cuerpo de las enfermedades</td><td>Prueba 2.pdf</td></tr><tr><td>Comer fruta y verdura tambien puede ayudarte a mantener un peso saludable, reducir el riesgo de padecer enfermedades cardiacas, derrames cerebrales y cancer, y mejorar tu salud y bienestar general</td><td>Comer frutas y verduras tambien puede ayudarte a mantener un peso saludable, reducir el riesgo de padecer enfermedades cardiacas, derrames cerebrales y cancer, y mejorar tu salud y bienestar general</td><td>Prueba 2.pdf</td></tr><tr><td>Comer frutas y verduras tiene muchos beneficios</td><td>Comer frutas y verduras tiene muchos beneficios</td><td>Prueba 2.pdf</td></tr><tr><td>Mejora de la salud del corazon: Las frutas y verduras son bajas en grasas saturadas y colesterol, y tienen un alto contenido en fibra</td><td>Mejora de la salud del corazon: Las frutas y verduras son bajas en grasas saturadas y colesterol, y tienen un alto contenido en fibra</td><td>Prueba 2.pdf</td></tr></table>	Oración plagiada	Oración original	Lugar donde se encontró	Las frutas y verduras son una parte esencial de una dieta sana	Las frutas y verduras son una parte importante de una dieta sana	Prueba 2.pdf	Estan repletas de vitaminas, minerales y antioxidantes que pueden ayudarle a proteger su cuerpo de las enfermedades	Estan repletas de vitaminas, minerales y antioxidantes que pueden ayudarle a proteger su cuerpo de las enfermedades	Prueba 2.pdf	Comer fruta y verdura tambien puede ayudarte a mantener un peso saludable, reducir el riesgo de padecer enfermedades cardiacas, derrames cerebrales y cancer, y mejorar tu salud y bienestar general	Comer frutas y verduras tambien puede ayudarte a mantener un peso saludable, reducir el riesgo de padecer enfermedades cardiacas, derrames cerebrales y cancer, y mejorar tu salud y bienestar general	Prueba 2.pdf	Comer frutas y verduras tiene muchos beneficios	Comer frutas y verduras tiene muchos beneficios	Prueba 2.pdf	Mejora de la salud del corazon: Las frutas y verduras son bajas en grasas saturadas y colesterol, y tienen un alto contenido en fibra	Mejora de la salud del corazon: Las frutas y verduras son bajas en grasas saturadas y colesterol, y tienen un alto contenido en fibra	Prueba 2.pdf
Oración plagiada	Oración original	Lugar donde se encontró																				
Las frutas y verduras son una parte esencial de una dieta sana	Las frutas y verduras son una parte importante de una dieta sana	Prueba 2.pdf																				
Estan repletas de vitaminas, minerales y antioxidantes que pueden ayudarle a proteger su cuerpo de las enfermedades	Estan repletas de vitaminas, minerales y antioxidantes que pueden ayudarle a proteger su cuerpo de las enfermedades	Prueba 2.pdf																				
Comer fruta y verdura tambien puede ayudarte a mantener un peso saludable, reducir el riesgo de padecer enfermedades cardiacas, derrames cerebrales y cancer, y mejorar tu salud y bienestar general	Comer frutas y verduras tambien puede ayudarte a mantener un peso saludable, reducir el riesgo de padecer enfermedades cardiacas, derrames cerebrales y cancer, y mejorar tu salud y bienestar general	Prueba 2.pdf																				
Comer frutas y verduras tiene muchos beneficios	Comer frutas y verduras tiene muchos beneficios	Prueba 2.pdf																				
Mejora de la salud del corazon: Las frutas y verduras son bajas en grasas saturadas y colesterol, y tienen un alto contenido en fibra	Mejora de la salud del corazon: Las frutas y verduras son bajas en grasas saturadas y colesterol, y tienen un alto contenido en fibra	Prueba 2.pdf																				
Caso 2	No es plagio: los textos son completamente diferentes	"Prueba 1"	"Prueba 2"	<p>Tópicos del texto: <i>cerebro, lóbulo, cuerpo, órgano, información</i></p> <p>Análisis de plagio</p> <p>Total de 0 plagios encontrados</p> <p>Porcentaje de plagio general: 0%</p> <table><tr><th>Oración plagiada</th><th>Oración original</th><th>Lugar donde se encontró</th></tr><tr><td></td><td></td><td></td></tr></table>	Oración plagiada	Oración original	Lugar donde se encontró															
Oración plagiada	Oración original	Lugar donde se encontró																				

Caso 3	Plagio parcial: los textos tienen similitud pero no son idénticos	"Prueba 1"	"Prueba 2"	<p>Tópicos del texto: <i>fruta, verdura, riesgo, salud, enfermedad</i></p> <p>Análisis de plagio</p> <p>Total de 21 plagios encontrados</p> <p>Porcentaje de plagio general: 80%</p> <table><tr><th>Oración plagiada</th><th>Oración original</th><th>Lugar donde se encontró</th></tr><tr><td>Las frutas y verduras son una parte importante de una dieta sana</td><td>Las frutas y verduras son una parte esencial de una dieta sana</td><td>Prueba1.pdf</td></tr><tr><td>Estan repletas de vitaminas, minerales y antioxidantes que pueden ayudarle a proteger su cuerpo de las enfermedades</td><td>Estan repletas de vitaminas, minerales y antioxidantes que pueden ayudarle a proteger su cuerpo de las enfermedades</td><td>Prueba1.pdf</td></tr><tr><td>Comer frutas y verduras tambien puede ayudarte a mantener un peso saludable, reducir el riesgo de padecer enfermedades cardiacas, derrames cerebrales y cancer, y mejorar tu salud y bienestar general</td><td>Comer fruta y verdura tambien puede ayudarte a mantener un peso saludable, reducir el riesgo de padecer enfermedades cardiacas, derrames cerebrales y cancer, y mejorar tu salud y bienestar general</td><td>Prueba1.pdf</td></tr><tr><td>Comer frutas y verduras tiene muchos beneficios</td><td>Comer frutas y verduras tiene muchos beneficios</td><td>Prueba1.pdf</td></tr><tr><td>Mejora de la salud del corazon: Las frutas y verduras son bajas en grasas saturadas y colesterol, y tienen un alto contenido en fibra</td><td>Mejora de la salud del corazon: Las frutas y verduras son bajas en grasas saturadas y colesterol, y tienen un alto contenido en fibra</td><td>Prueba1.pdf</td></tr><tr><td>Esta las convierta en una</td><td>Esta las convierta en una</td><td></td></tr></table>	Oración plagiada	Oración original	Lugar donde se encontró	Las frutas y verduras son una parte importante de una dieta sana	Las frutas y verduras son una parte esencial de una dieta sana	Prueba1.pdf	Estan repletas de vitaminas, minerales y antioxidantes que pueden ayudarle a proteger su cuerpo de las enfermedades	Estan repletas de vitaminas, minerales y antioxidantes que pueden ayudarle a proteger su cuerpo de las enfermedades	Prueba1.pdf	Comer frutas y verduras tambien puede ayudarte a mantener un peso saludable, reducir el riesgo de padecer enfermedades cardiacas, derrames cerebrales y cancer, y mejorar tu salud y bienestar general	Comer fruta y verdura tambien puede ayudarte a mantener un peso saludable, reducir el riesgo de padecer enfermedades cardiacas, derrames cerebrales y cancer, y mejorar tu salud y bienestar general	Prueba1.pdf	Comer frutas y verduras tiene muchos beneficios	Comer frutas y verduras tiene muchos beneficios	Prueba1.pdf	Mejora de la salud del corazon: Las frutas y verduras son bajas en grasas saturadas y colesterol, y tienen un alto contenido en fibra	Mejora de la salud del corazon: Las frutas y verduras son bajas en grasas saturadas y colesterol, y tienen un alto contenido en fibra	Prueba1.pdf	Esta las convierta en una	Esta las convierta en una	
Oración plagiada	Oración original	Lugar donde se encontró																							
Las frutas y verduras son una parte importante de una dieta sana	Las frutas y verduras son una parte esencial de una dieta sana	Prueba1.pdf																							
Estan repletas de vitaminas, minerales y antioxidantes que pueden ayudarle a proteger su cuerpo de las enfermedades	Estan repletas de vitaminas, minerales y antioxidantes que pueden ayudarle a proteger su cuerpo de las enfermedades	Prueba1.pdf																							
Comer frutas y verduras tambien puede ayudarte a mantener un peso saludable, reducir el riesgo de padecer enfermedades cardiacas, derrames cerebrales y cancer, y mejorar tu salud y bienestar general	Comer fruta y verdura tambien puede ayudarte a mantener un peso saludable, reducir el riesgo de padecer enfermedades cardiacas, derrames cerebrales y cancer, y mejorar tu salud y bienestar general	Prueba1.pdf																							
Comer frutas y verduras tiene muchos beneficios	Comer frutas y verduras tiene muchos beneficios	Prueba1.pdf																							
Mejora de la salud del corazon: Las frutas y verduras son bajas en grasas saturadas y colesterol, y tienen un alto contenido en fibra	Mejora de la salud del corazon: Las frutas y verduras son bajas en grasas saturadas y colesterol, y tienen un alto contenido en fibra	Prueba1.pdf																							
Esta las convierta en una	Esta las convierta en una																								
Caso 4	Textos similares: los textos tienen una alta similitud pero no son plagio	"Prueba 1"	"Prueba 2"	<p>Tópicos del texto: <i>cerebro, lobulo, organo, cuerpo, informacion</i></p> <p>Análisis de plagio</p> <p>Total de 5 plagios encontrados</p> <p>Porcentaje de plagio general: 16%</p> <table><tr><th>Oración plagiada</th><th>Oración original</th><th>Lugar donde se encontró</th></tr><tr><td>El cerebro humano es un organo altamente complejo y fundamental en nuestro cuerpo</td><td>El cerebro humano es un organo complejo y fascinante</td><td>Prueba 3.pdf</td></tr><tr><td>Es responsable de una amplia gama de funciones, desde el procesamiento de pensamientos y emociones hasta el control de nuestros movimientos y sentidos</td><td>Es responsable de todo, desde nuestros pensamientos y emociones hasta nuestros movimientos y sentidos</td><td>Prueba 3.pdf</td></tr><tr><td>Aqui te presento algunos datos adicionales sobre el cerebro:</td><td>He aqui algunos datos adicionales sobre el cerebro:</td><td>Prueba 3.pdf</td></tr><tr><td>El cerebro tiene un peso aproximado de 1,5 kg, representando alrededor del 2% de nuestro peso corporal</td><td>El cerebro pesa alrededor de 1,5 kg y constituye aproximadamente el 2% del peso corporal</td><td>Prueba 3.pdf</td></tr><tr><td>A lo largo de nuestra vida, el cerebro continua cambiando y adaptandose</td><td>El cerebro sigue cambiando y adaptandose a lo largo de nuestra vida</td><td>Prueba 3.pdf</td></tr></table>	Oración plagiada	Oración original	Lugar donde se encontró	El cerebro humano es un organo altamente complejo y fundamental en nuestro cuerpo	El cerebro humano es un organo complejo y fascinante	Prueba 3.pdf	Es responsable de una amplia gama de funciones, desde el procesamiento de pensamientos y emociones hasta el control de nuestros movimientos y sentidos	Es responsable de todo, desde nuestros pensamientos y emociones hasta nuestros movimientos y sentidos	Prueba 3.pdf	Aqui te presento algunos datos adicionales sobre el cerebro:	He aqui algunos datos adicionales sobre el cerebro:	Prueba 3.pdf	El cerebro tiene un peso aproximado de 1,5 kg, representando alrededor del 2% de nuestro peso corporal	El cerebro pesa alrededor de 1,5 kg y constituye aproximadamente el 2% del peso corporal	Prueba 3.pdf	A lo largo de nuestra vida, el cerebro continua cambiando y adaptandose	El cerebro sigue cambiando y adaptandose a lo largo de nuestra vida	Prueba 3.pdf			
Oración plagiada	Oración original	Lugar donde se encontró																							
El cerebro humano es un organo altamente complejo y fundamental en nuestro cuerpo	El cerebro humano es un organo complejo y fascinante	Prueba 3.pdf																							
Es responsable de una amplia gama de funciones, desde el procesamiento de pensamientos y emociones hasta el control de nuestros movimientos y sentidos	Es responsable de todo, desde nuestros pensamientos y emociones hasta nuestros movimientos y sentidos	Prueba 3.pdf																							
Aqui te presento algunos datos adicionales sobre el cerebro:	He aqui algunos datos adicionales sobre el cerebro:	Prueba 3.pdf																							
El cerebro tiene un peso aproximado de 1,5 kg, representando alrededor del 2% de nuestro peso corporal	El cerebro pesa alrededor de 1,5 kg y constituye aproximadamente el 2% del peso corporal	Prueba 3.pdf																							
A lo largo de nuestra vida, el cerebro continua cambiando y adaptandose	El cerebro sigue cambiando y adaptandose a lo largo de nuestra vida	Prueba 3.pdf																							

6.1.2. Iteración 2:

Entrenar el modelo utilizando Scikit-Learn.

6.1.2.1. Fase de Análisis

Actividad 1: Análisis de los datos.

Se realizó un análisis de los datos que se utilizaran en la detección de plagio, centrándonos en aspectos como el formato de los documentos, la estructura, la longitud de los textos y la distribución de los párrafos. Este análisis permitió obtener una comprensión detallada de las características de los datos y su impacto en el proceso de entrenamiento. Se examinó el formato de los documentos para determinar si eran archivos de texto plano o en otro formato específico. Además, se evaluaron la estructura y la longitud de los textos para identificar patrones comunes en la detección de plagio. También se analizó la distribución de los párrafos para comprender la organización del contenido y su relevancia para la detección de similitudes. Estos conocimientos fueron fundamentales para planificar y diseñar adecuadamente el procesamiento de entrenamiento, teniendo en cuenta las particularidades de los datos utilizados en el sistema de detección de plagio.

6.1.2.2. Fase de Diseño

Actividades 2: Generación del Dataset

En esta actividad la creación del dataset para detectar plagio, lo que implicó la recopilación y creación de conjuntos de datos donde incluyen una variedad de textos originales y potencialmente plagiados. Se obtuvieron diversos textos extraídos del repositorio DSpace de la Universidad Nacional del Loja (UNL) de la carrera de Computación. Con el fin de mejorar la efectividad del dataset, se realizaron modificaciones en algunos textos, tanto copiándolos textualmente como parafraseándolos. Se generaron dos conjuntos diferentes de datos: uno que consistió en un 80% de documentos parafraseados y otro que representó el 20%. Esto permitió contar con una mayor variedad de valores de prueba en el dataset. Para visualizar el Dataset de Documentos (véase el anexo 4)

6.1.2.3. Fase de Codificación

Actividad 1: Importación de Librerías y Módulos.

Las librerías que se utilizaron para la codificación de la red neuronal son: Scikit-learn, numpy, googlesearch, bs4 (BeautifulSoup), nltk, TfidfVectorizer (de sklearn.feature_extraction.text), metodos_de_similitud, helper y procesamiento_de_archivos. Para mas informacion de estas librerias (vease en el marco teorico).

- **metodos_de_similitud:** Módulo personalizado que contiene métodos para calcular la similitud entre textos.

- **helper:** Módulo personalizado que contiene funciones y variables de apoyo para el procesamiento de archivos y modelos.
- **procesamiento_de_archivos:** Módulo personalizado que contiene funciones para la limpieza y procesamiento de archivos de texto.
- **TfidfVectorizer (de sklearn.feature_extraction.text):** Implementa la técnica de vectorización TF-IDF para convertir texto en características numéricas.

Actividad 2: Carga y preparación de los Datos.

Los datos de entrenamiento y prueba en formato txt fueron leídos y procesados. Para procesar los datos, se cargaron archivos de texto en formato txt en el sistema. Luego, se utilizaron técnicas de procesamiento de texto como la tokenización (división del texto en palabras o tokens), la lematización (reducción de las palabras a su forma base) y la normalización (ajuste de las palabras a su forma original).

Actividad 3: Codificar la Red Neuronal Siamesa.

En esta actividad, se procedió a implementar los algoritmos y estructuras de datos necesarios para construir la arquitectura de la red neuronal siamesa. Esto involucró la lógica detrás del análisis del dataset generado anteriormente, que permitió obtener los pares de documentos generados, cada uno acompañado de su respectiva similitud entre pares. Además, se llevó a cabo el análisis de los pares de textos para obtener el conjunto de entrenamiento necesario para la red neuronal. Asimismo, se realizó la definición de las capas requeridas para la correcta operación de la red neuronal, teniendo en cuenta los parámetros específicos de cada rama de cada capa. Este proceso abarcó la configuración de las capas de entrada, procesamiento y salida, asegurando que estuvieran correctamente conectadas y configuradas para la tarea de detección de similitudes entre textos. Todo esto fue realizado con el objetivo de construir una arquitectura de red neuronal siamesa que fuera capaz de cumplir con la tarea de detección de similitudes entre textos.

Descripción de las funciones Principales

- La función generar_pares_textos:** Esta función recibe una ruta pares_textos_path, que se guardará en un archivo de texto con los pares de textos y sus similitudes. Además, se le envía una lista de documentos que incluye los textos de los documentos que se deben comparar. La función vectoriza los textos de los documentos mediante la técnica TF-IDF. Luego, determina la similitud del coseno de cada par de textos y almacena los resultados en el conjunto de resultados. Por último guarda los pares de textos y sus equivalentes en el archivo pares_textos.txt.

La elección de un vectorizador TF-IDF con una **longitud máxima de 10.000** y un rango de **n-gramas de 1 a 2** en la función de vectorización lambda se justifica por el equilibrio entre precisión y generalización que proporciona. Al establecer una longitud máxima más pequeña, se evita la captura de detalles precisos, pero poco relevantes que podrían conducir al sobreajuste del modelo. Esto garantiza que el modelo se ajuste de manera más adecuada a los datos de entrenamiento y generalice mejor a nuevos datos. Además, esta elección también tiene en cuenta consideraciones de eficiencia computacional al limitar la dimensionalidad de los vectores y captura información contextual relevante mediante la inclusión de bigramas. En resumen, la elección de estos parámetros busca obtener un equilibrio óptimo entre precisión, generalización y eficiencia computacional en el proceso de vectorización del texto.

```
def generar_pares_textos(pares_textos_path, documentos): #fuera
    #cuidado, posible implementación de detección de plagio para proseguir
    sample_files = [doc.nombre + doc.extension for doc in documentos]
    sample_contents = ["".join(doc.texto) for doc in documentos]
    sample_contents_lemmatized = sample_contents

    vectorize = lambda Text: TfidfVectorizer(max_features=10000, ngram_range=(1, 2), sublinear_tf=True, smooth_idf=True).fit_transform(Text).toarray()
    similarity = lambda doc1, doc2: cosine_similarity([doc1, doc2])

    vectors = vectorize(sample_contents_lemmatized)
    s_vectors = list(zip(sample_files, vectors))

    results = set()
    for sample_a, text_vector_a in s_vectors:
        new_vectors = s_vectors.copy()
        current_index = new_vectors.index((sample_a, text_vector_a))
        del new_vectors[current_index]
        for sample_b, text_vector_b in new_vectors:
            sim_score = similarity(text_vector_a, text_vector_b)[0][1]
            sample_pair = sorted((sample_a, sample_b))
            score = sample_pair[0], sample_pair[1], sim_score
            results.add(score)

    with open(os.path.join(pares_textos_path, "pares_textos.txt"), 'w') as f:
        for data in results:
            print(data, file=f)
```

Figura 18. Función para generar pares de textos.

- b. **La función leer_pares_textos:** Lee pares de textos y sus similitudes desde el archivo pares_textos.txt, que se encuentra en la ruta pares_textos_path. Se debe ir sobre cada línea del archivo, realizar el procesamiento necesario para extraer los textos y la similitud, y luego agregarlos a las listas train_text1, train_text2 y train_similarity, respectivamente. Luego, vuelve a enviar estas listas.


```
def leer_pares_textos(pares_textos_path, documentos):
    text_pairs = []
    train_text1 = []
    train_text2 = []
    train_similarity = []
    with open(pares_textos_path + "pares_textos.txt", "r") as file: #ojo
        for line in file:
            line = line.strip().strip('()').replace("'", "")
            texto1, texto2, sim = line.split(", ")
            doc1 = texto1
            doc2 = texto2
            similarity = float(sim)
            text_pairs.append((doc1, doc2, similarity))

    for pair in text_pairs:
        doc1 = pair[0]
        doc2 = pair[1]
        similarity = pair[2]

        # Leer el texto en los documentos
        documento1 = documento_por_nombre(documentos, doc1)
        documento2 = documento_por_nombre(documentos, doc2)

        if documento1 and documento2:
            # Ambos documentos encontrados, se pueden acceder a sus atributos
            text1 = documento1.texto
            text2 = documento2.texto

            train_text1.append(text1)
            train_text2.append(text2)
            train_similarity.append(similarity)
        else:
            # Alguno de los documentos no se encontró, manejar el caso en consecuencia
            print(f"Error: No se encontró el documento para la pareja {doc1} y {doc2}")

    train_similarity = np.array(train_similarity)

    return train_text1, train_text2, train_similarity
```

Figura 19. Función para la lectura de pares de textos.

- c. **La función generar_modelo_entrenado:** Responsable de generar y entrenar el modelo utilizando los pares de textos que están disponibles. Para generar los pares de textos y calcular sus similitudes, primero llama a la función generar_pares_textos. Para obtener listas de textos y similitudes, utilice la función leer_pares_textos. Luego, utiliza la clase Tokenizer de Keras para procesar los textos y convertirlos en secuencias numéricas. A continuación, utilice la función pad_sequences para rellenar las secuencias para que tengan la misma longitud. Definió el modelo de una red neuronal siamesa con capas de inserción, concatenación, salida y densidad. El optimizador Adam y la función de pérdida de crossentropía binaria se utilizan para compilar el modelo. Por último, entrena el modelo con los datos de entrenamiento train_data1 y train_data2 y las etiquetas de similitud train_similarity.

Justificación para los valores específicos utilizados en el código

max_words : El valor máximo de palabras a tener en cuenta en el vocabulario es de 10.000. La reducción del número de palabras reduce la dimensionalidad de los datos y aumenta la eficiencia computacional. Además, se selecciona un valor lo suficientemente grande para incluir una amplia gama de palabras pertinentes en el texto.

embedding_dim: El parámetro embedding_dim se refiere a la dimensión del espacio de incrustación utilizado para representar las palabras. Un valor de 100 indica que cada palabra se representará en un espacio de 100 dimensiones. Esta dimensión suele ser suficiente para capturar características semánticas importantes en el texto. Entonces que sucedería si se aumenta o disminuye este valor:

- Si se aumenta el valor de `embedding_dim`, un valor demasiado grande puede aumentar la complejidad computacional y requerir más memoria para almacenar los vectores de incrustación.
- Si se reduce el valor de `embedding_dim`. Esto podría resultar en una pérdida de información y una disminución en la capacidad de capturar elementos semánticos complejos en el texto.

Es importante encontrar un equilibrio entre la capacidad de representación del modelo y los recursos computacionales disponibles para lograr un rendimiento óptimo.

hidden_units: Representa el número de unidades en la capa oculta de la red neuronal. Se selecciona un valor de 128 para equilibrar la capacidad de aprendizaje del modelo y la complejidad computacional. Puede modificarse según el tamaño y las necesidades del conjunto de datos. Entonces que sucedería si se aumenta o disminuye este valor:

- Incrementar el número de unidades en la capa oculta puede causar sobreajuste, perdiendo la capacidad de generalizar en nuevos datos. Esto se debe a la mayor complejidad y el riesgo de ajuste excesivo, lo que afecta negativamente el rendimiento del modelo. Es esencial encontrar un equilibrio adecuado en la elección del número de unidades para evitar estos problemas y permitir un mejor rendimiento en los datos.
- Reducir el valor puede disminuir la capacidad de aprendizaje y la capacidad de capturar relaciones más complejas en los datos. La elección adecuada de `hidden_units` depende de la complejidad del problema y de la cantidad de datos disponibles.

dropout_rate: Muestra la cantidad de unidades que se desactivan aleatoriamente durante el entrenamiento para evitar el sobreajuste. El valor de 0,2 indica que en cada paso de entrenamiento se desactiva el 20% de las unidades. Esto ayuda a regularizar el modelo y mejorar su generalización. Entonces que sucedería si se aumenta o disminuye este valor:

- Si se aumenta puede llevar a un subajuste, donde el modelo no aprende lo suficiente y no se ajusta adecuadamente a los datos de entrenamiento.
- Si se disminuye se desactivan menos neuronas durante el entrenamiento, lo que aumenta la dependencia de neuronas individuales y aumenta la probabilidad de que el modelo se sobreajuste.

optimizer : El optimizador Adam se utiliza para ajustar los pesos del modelo durante el entrenamiento. La tasa de ajuste de los pesos está controlada por un rendimiento (lr) de 0.001. Entonces que sucedería si se aumenta o disminuye este valor:

- Si aumenta el proceso de entrenamiento puede volverse inestable y los resultados del modelo pueden ser poco confiables o deficientes.
- Si disminuye puede hacer que el proceso de entrenamiento sea más lento. El modelo necesitará más iteraciones y tiempo para ajustar sus parámetros y aprender de los datos de entrenamiento. Esto no sería conveniente cuando se este trabajando con datos muy grandes o cuando se tenga limitaciones de tiempo.

epochs: El número de veces que el modelo recorrerá todo el conjunto de entrenamiento durante el proceso de entrenamiento, el valor de 20 se selecciona como un número suficiente de iteraciones para que el modelo aprenda y ajuste sus parámetros, pero sin gastar demasiado tiempo en el proceso.

batch_size: Representa el número de muestras de entrenamiento que se procesan antes de que se actualicen los parámetros del modelo. En este caso, se utiliza un tamaño de lote de 32, lo que significa que se procesarán 32 muestras a la vez antes de actualizar los pesos del modelo. El tamaño del lote se determina en función de la eficiencia computacional y la capacidad de memoria.

```

def generar_modelo_entrenado(pares_textos_path):

    # Ingresar el numero maximo de palabras a considerar
    max_words = 10000
    train_text1 = []
    train_text2 = []
    train_similarity = []
    documentos = archivos_referencia_limpios
    generar_pares_textos(pares_textos_path, documentos)
    train_text1, train_text2, train_similarity = leer_pares_textos(pares_textos_path, documentos)

    tokenizer = Tokenizer(num_words=max_words)
    tokenizer.fit_on_texts(train_text1 + train_text2)

    train_sequences1 = tokenizer.texts_to_sequences(train_text1)
    train_sequences2 = tokenizer.texts_to_sequences(train_text2)

    train_data1 = pad_sequences(train_sequences1)
    train_data2 = pad_sequences(train_sequences2)

    embedding_dim = 100 #200
    hidden_units = 128 #256
    dropout_rate = 0.2

    input1 = Input(shape=(train_data1.shape[1],))
    input2 = Input(shape=(train_data2.shape[1],))

    shared_embedding_layer = Embedding(max_words, embedding_dim)

    embedded1 = shared_embedding_layer(input1)
    embedded2 = shared_embedding_layer(input2)

    flattened1 = Flatten()(embedded1)
    flattened2 = Flatten()(embedded2)

    concatenated = Concatenate()([flattened1, flattened2])
    dropout = Dropout(dropout_rate)(concatenated)
    output = Dense(hidden_units, activation='relu')(dropout)
    output = Dense(1, activation='sigmoid')(output)

    model = Model(inputs=[input1, input2], outputs=output)
    optimizer = Adam(lr=0.001)
    model.compile(loss='binary_crossentropy', optimizer=optimizer, metrics=['accuracy'])

    model.fit([train_data1, train_data2], train_similarity, epochs=20, batch_size=32)

    modelo_entrenado.append((model, tokenizer, train_data1, train_data2))

```

Figura 20. Función para generar el modelo entrenado.

- d. La función calcular_similitud:** Utiliza el modelo entrenado para encontrar qué tan similares son dos textos. Tiene dos textos recibidos: texto1 y texto2. Primero, encuentra el modelo entrenado, el tokenizer y los datos de entrenamiento en la lista modelo_entrenado. Luego, procesa los datos de entrada text1_data y text2_data utilizando el tokenizer y el padding. Por último, el modelo se utiliza para predecir la similitud entre los textos y devolver el valor de similitud.

```
def calcular_similitud( text1, text2):
    model, tokenizer, train_data1, train_data2 = modelo_entrenado[0]
    text1_sequence = tokenizer.texts_to_sequences([text1])
    text2_sequence = tokenizer.texts_to_sequences([text2])
    text1_data = pad_sequences(text1_sequence, maxlen=train_data1.shape[1])
    text2_data = pad_sequences(text2_sequence, maxlen=train_data2.shape[1])

    similarity = model.predict([text1_data, text2_data])[0][0]
    return similarity
```

Figura 21. Función para calcular la similitud.

Descripción de las Capas Codificadas

- **Capa de entrada:** Recibe los pares de texto que se deben comparar.
- **Capa de Embedding:** Para representar las palabras de manera numérica, la capa de embedding convierte las secuencias de palabras en vectores densos de tamaño fijo.
- **Capa de Flatten:** Aplana los vectores de embedding para que puedan ser concatenados.
- **Capa de Concatenación:** Concatena los vectores de inserción de los dos textos en un vector único.
- **Capa Dropout:** Utiliza una técnica de regularización conocida como salida para evitar el sobreajuste y mejorar la generalización del modelo.
- **Capas densas (Fully Connected):** Aplica capas densas (totalmente conectadas) para realizar la clasificación final de la similitud entre los textos.
- **Capa de salida:** La capa de salida produce el resultado final del modelo, que es la predicción de la similitud entre los textos. Dado que se trata de un problema de clasificación binaria donde se busca predecir la similitud (valor entre 0 y 1) entre los textos, la función de activación utilizada en la última capa es una función sigmoide.

6.1.2.4. Fase de Pruebas

Actividad 1: Utilizar el dataset generado para las pruebas.

Para el dataset se utilizaron múltiples documentos obtenidos de internet, que serían en primera instancia los documentos con los que se iniciarán las pruebas unitarias. Para hacer más sencillo el control de los documentos se los llamó documento (1), documento (2), etc. Usando hasta una cantidad de 35 documentos inicialmente para generar nuestro dataset inicial. Una vez que se obtuvo un dataset inicial, se procedió a obtener los pares de textos de todos los documentos. Los pares de textos representan una combinación de 2 elementos sin repetición en la que se agrega adicionalmente la similitud que existe entre ambos documentos.

Para obtener la similitud inicial se hizo uso de la similitud de coseno utilizando los métodos que proporciona scikit-learn, dándonos así la base del entrenamiento del modelo de red neuronal a usar.

```

def check_plagiarism_and_save_results(documentos):
    sample_files = [doc.nombre + doc.extension for doc in documentos]
    sample_contents = [doc.texto for doc in documentos]
    sample_contents_lemmatized = sample_contents

    vectorize = lambda Text: TfidfVectorizer(max_features=10000, ngram_range=(1, 2), sublinear_tf=True, smooth_idf=True).fit_transform(Text).toarray()
    similarity = lambda doc1, doc2: cosine_similarity([doc1, doc2])

    vectors = vectorize(sample_contents_lemmatized)
    s_vectors = list(zip(sample_files, vectors))

    results = set()
    for sample_a, text_vector_a in s_vectors:
        new_vectors = s_vectors.copy()
        current_index = new_vectors.index((sample_a, text_vector_a))
        del new_vectors[current_index]
        for sample_b, text_vector_b in new_vectors:
            sim_score = similarity(text_vector_a, text_vector_b)[0][1]
            sample_pair = sorted((sample_a, sample_b))
            score = sample_pair[0], sample_pair[1], sim_score
            results.add(score)

    oraciones_list = texto_oraciones.split('\n')
    vectorize_oraciones = vectorize(oraciones_list)
    pair_similarities = []
    for i in range(len(oraciones_list)):
        for j in range(i+1, len(oraciones_list)):
            sim_score = similarity(vectorize_oraciones[i], vectorize_oraciones[j])[0][1]
            pair_similarity = oraciones_list[i], oraciones_list[j], sim_score
            pair_similarities.append(pair_similarity)

    with open('pares_oraciones.txt', 'w', encoding='utf-8') as f:
        for data in pair_similarities:
            print(data, file=f)

    with open('pares_textos.txt', 'w') as f:
        for data in results:
            print(data, file=f)

```

Figura 22. Similitud usando el método del coseno.

Una vez se tenga los pares de textos, todos con su respectiva similitud como se puede ver en la imagen (siguiente), se podrá comenzar a darle los datos a la red neuronal y generar su modelo.

```

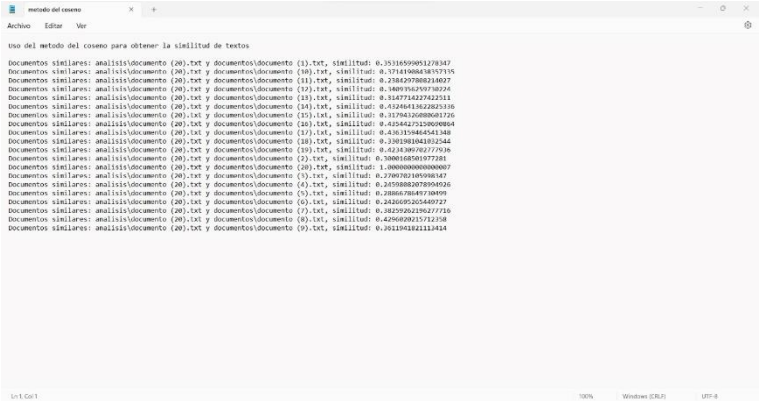
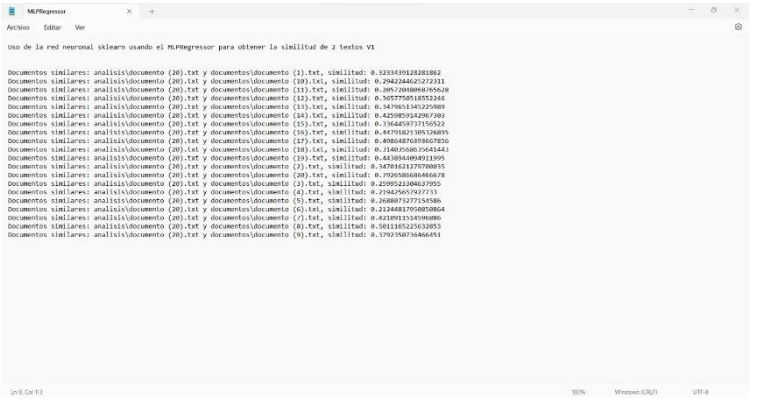
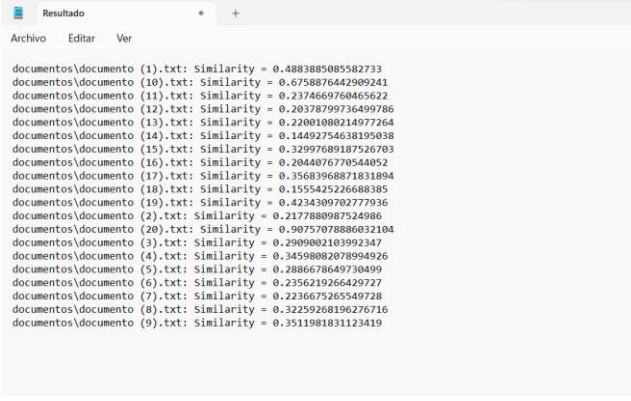
('documento (13).txt', 'documento (18).txt', 0.07163613386893444)
('documento (20).txt', 'documento (7).txt', 0.06813165699637105)
('documento (17) - copia.txt', 'documento (3).txt', 0.04559440026160893)
('documento (13).txt', 'documento (2).txt', 0.035944485278378884)
('documento (12) - copia.txt', 'documento (5).txt', 0.0874691353240711)
('documento (14).txt', 'documento (9) - copia.txt', 0.08053174867897675)
('documento (18).txt', 'documento (19).txt', 0.08761110799421545)
('documento (19).txt', 'documento (3).txt', 0.04714171578362813)
('documento (16).txt', 'documento (19).txt', 0.06940164242705148)
('documento (13).txt', 'documento (6).txt', 0.03935871469504841)
('documento (12).txt', 'documento (7).txt', 0.05623732759478666)
('documento (14) - copia.txt', 'documento (18).txt', 0.06759342710977195)
('documento (4).txt', 'documento (9) - copia.txt', 0.05971970278226339)
('documento (4) - copia.txt', 'documento (4).txt', 0.9999999999999999)
('documento (1).txt', 'documento (7).txt', 0.0740428610904872)
('documento (19).txt', 'documento (2) - copia.txt', 0.049618287131270145)
('documento (13) - copia.txt', 'documento (9) - copia.txt', 0.15676672329640004)
('documento (1).txt', 'documento (9).txt', 0.09544230715012683)
('documento (7).txt', 'documento (9).txt', 0.12212994317705465)
('documento (11) - copia.txt', 'documento (3) - copia.txt', 0.03480164824889194)
('documento (13).txt', 'documento (7) - copia.txt', 0.13412721206806336)
('documento (16).txt', 'documento (7).txt', 0.05466276137609114)
('documento (15).txt', 'documento (3) - copia.txt', 0.039153543104604595)
('documento (2) - copia.txt', 'documento (9) - copia.txt', 0.033644293980617475)
('documento (15).txt', 'documento (7) - copia.txt', 0.07954406951479268)
('documento (20).txt', 'documento (5).txt', 0.05971325011236662)
('documento (4).txt', 'documento (8).txt', 0.05626733206795698)
('documento (1).txt', 'documento (11).txt', 0.07966596089156764)
('documento (10).txt', 'documento (16).txt', 0.1542828876431444)
('documento (13) - copia.txt', 'documento (4) - copia.txt', 0.06598479059658444)
('documento (14) - copia.txt', 'documento (8).txt', 0.0799288440240753)
('documento (19).txt', 'documento (7) - copia.txt', 0.13795567485883736)
('documento (1) - copia.txt', 'documento (7).txt', 0.07740428610904872)
('documento (15).txt', 'documento (17).txt', 0.0017664940533933)
('documento (13).txt', 'documento (9).txt', 0.15676672329640004)
('documento (12).txt', 'documento (17) - copia.txt', 0.0097639596542496)
('documento (14).txt', 'documento (6) - copia.txt', 0.028069406347084038)
('documento (12).txt', 'documento (8) - copia.txt', 0.08376573984597077)
('documento (4) - copia.txt', 'documento (5) - copia.txt', 0.03183545100675796)
('documento (8).txt', 'documento (9) - copia.txt', 0.08436232833530698)

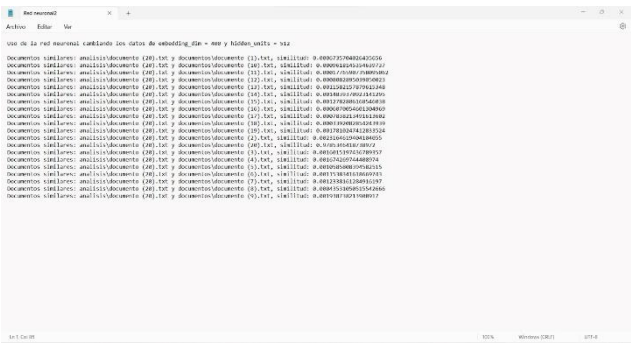
```

Figura 23. Documento con los pares textos.

Una vez que toda la información de pares textos pueda ser usada para la red neuronal, se hicieron diferentes tipos de pruebas para mostrar la efectividad de los datos que se muestra en la siguiente tabla:

Tabla 11. Casos de Prueba para la Iteración 2.

Caso de prueba	Descripción	Resultado esperado
Caso 1	Generar los valores de similitud usando el método del coseno	 <pre> uso del método del coseno para obtener la similitud de textos Documentos similares: análisis\documento (20).txt y documentos\documento (1).txt, similitud: 0.3516599051278347 Documentos similares: análisis\documento (20).txt y documentos\documento (10).txt, similitud: 0.3714190841017194 Documentos similares: análisis\documento (20).txt y documentos\documento (11).txt, similitud: 0.2184297808218027 Documentos similares: análisis\documento (20).txt y documentos\documento (12).txt, similitud: 0.340793279786294 Documentos similares: análisis\documento (20).txt y documentos\documento (13).txt, similitud: 0.314771422742511 Documentos similares: análisis\documento (20).txt y documentos\documento (14).txt, similitud: 0.412461182262518 Documentos similares: análisis\documento (20).txt y documentos\documento (15).txt, similitud: 0.3178150680601726 Documentos similares: análisis\documento (20).txt y documentos\documento (16).txt, similitud: 0.4154847310008084 Documentos similares: análisis\documento (20).txt y documentos\documento (17).txt, similitud: 0.431375046441348 Documentos similares: análisis\documento (20).txt y documentos\documento (18).txt, similitud: 0.330198184912544 Documentos similares: análisis\documento (20).txt y documentos\documento (19).txt, similitud: 0.4213136792177936 Documentos similares: análisis\documento (20).txt y documentos\documento (2).txt, similitud: 0.3008168501877381 Documentos similares: análisis\documento (20).txt y documentos\documento (3).txt, similitud: 0.2008008000000007 Documentos similares: análisis\documento (20).txt y documentos\documento (4).txt, similitud: 0.2790782189986147 Documentos similares: análisis\documento (20).txt y documentos\documento (5).txt, similitud: 0.2459082079509506 Documentos similares: análisis\documento (20).txt y documentos\documento (6).txt, similitud: 0.288678649740099 Documentos similares: análisis\documento (20).txt y documentos\documento (7).txt, similitud: 0.28269525469727 Documentos similares: análisis\documento (20).txt y documentos\documento (8).txt, similitud: 0.3827262126277710 Documentos similares: análisis\documento (20).txt y documentos\documento (9).txt, similitud: 0.4256620815712358 Documentos similares: análisis\documento (20).txt y documentos\documento (9).txt, similitud: 0.3511981831123419 </pre>
Caso 2	Generar los valores de similitud utilizando una red neuronal Multi-Capa	 <pre> uso de la red neuronal sklearn usando el MLRegressor para obtener la similitud de 2 textos v1 Documentos similares: análisis\documento (20).txt y documentos\documento (1).txt, similitud: 0.323349128338862 Documentos similares: análisis\documento (20).txt y documentos\documento (10).txt, similitud: 0.284246425272213 Documentos similares: análisis\documento (20).txt y documentos\documento (11).txt, similitud: 0.295720868785648 Documentos similares: análisis\documento (20).txt y documentos\documento (12).txt, similitud: 0.507759518552266 Documentos similares: análisis\documento (20).txt y documentos\documento (13).txt, similitud: 0.347951343270086 Documentos similares: análisis\documento (20).txt y documentos\documento (14).txt, similitud: 0.425792514287399 Documentos similares: análisis\documento (20).txt y documentos\documento (15).txt, similitud: 0.33642597155552 Documentos similares: análisis\documento (20).txt y documentos\documento (16).txt, similitud: 0.457162138570885 Documentos similares: análisis\documento (20).txt y documentos\documento (17).txt, similitud: 0.4906487493667826 Documentos similares: análisis\documento (20).txt y documentos\documento (18).txt, similitud: 0.346756815464443 Documentos similares: análisis\documento (20).txt y documentos\documento (19).txt, similitud: 0.478162117708055 Documentos similares: análisis\documento (20).txt y documentos\documento (2).txt, similitud: 0.3478162117708055 Documentos similares: análisis\documento (20).txt y documentos\documento (3).txt, similitud: 0.431045494911395 Documentos similares: análisis\documento (20).txt y documentos\documento (4).txt, similitud: 0.3478162117708055 Documentos similares: análisis\documento (20).txt y documentos\documento (5).txt, similitud: 0.793086846466478 Documentos similares: análisis\documento (20).txt y documentos\documento (6).txt, similitud: 0.23562192566420727 Documentos similares: análisis\documento (20).txt y documentos\documento (7).txt, similitud: 0.218465174277131 Documentos similares: análisis\documento (20).txt y documentos\documento (8).txt, similitud: 0.318481759870864 Documentos similares: análisis\documento (20).txt y documentos\documento (9).txt, similitud: 0.3511981831123419 Documentos similares: análisis\documento (20).txt y documentos\documento (9).txt, similitud: 0.3511981831123419 </pre>
Caso 3	Generar los valores de similitud usando la red neuronal	 <pre> Resultado documentos\documento (1).txt: Similarity = 0.4883885885882733 documentos\documento (10).txt: Similarity = 0.6758876442909241 documentos\documento (11).txt: Similarity = 0.2374669760465622 documentos\documento (12).txt: Similarity = 0.20378799736499786 documentos\documento (13).txt: Similarity = 0.22001080214977264 documentos\documento (14).txt: Similarity = 0.14492754638195038 documentos\documento (15).txt: Similarity = 0.32997689187526703 documentos\documento (16).txt: Similarity = 0.2044076779544052 documentos\documento (17).txt: Similarity = 0.35683968871831894 documentos\documento (18).txt: Similarity = 0.1555425226688385 documentos\documento (19).txt: Similarity = 0.4234309702777936 documentos\documento (2).txt: Similarity = 0.2177880987524986 documentos\documento (20).txt: Similarity = 0.90757078886032104 documentos\documento (3).txt: Similarity = 0.2909002103992347 documentos\documento (4).txt: Similarity = 0.34598082078994926 documentos\documento (5).txt: Similarity = 0.2886678649730499 documentos\documento (6).txt: Similarity = 0.23562192566420727 documentos\documento (7).txt: Similarity = 0.2236675265549728 documentos\documento (8).txt: Similarity = 0.32259268196276716 documentos\documento (9).txt: Similarity = 0.3511981831123419 </pre>

Caso 4	Generar los valores de similitud usando la red neuronal cambiando las constantes de la red neuronal	
--------	---	--

6.2. Objetivo 2:

Diseñar el software antiplagio utilizando Django y aplicando del modelo de software 4+1.

6.3. Objetivo 3:

Evaluar la ejecución del software antiplagio, dentro de la carrera de Ingeniería en Computación, en los proyectos PIC.

7. Discusión

8. Conclusiones

9. Recomendaciones

10. Bibliografía

- [1] J. Fernando Sánchez-Vega, M. Montes Y Gómez, and P. Rosso, "Identificación de plagio parafraseado incorporando estructura, sentido y estilo de los textos," Feb. 2016.
- [2] R. Ramírez Bacca, H. David, and J. Patiño, "Plagio y 'auto-plagio'. Una reflexión Plagiarism and Self-plagiarism. A Reflexion," Jul. 2016.
- [3] U. Y. Sociedad, E. Enrique, and E. Freire, "EL PLAGIO UN FLAGELO EN EL ÁMBITO ACADÉMICO ECUATORIANO," May 2020.
- [4] Sandra Timal López and Francisco Sánchez Espinoza, "El plagio en el contexto del derecho de autor," Universidad Autónoma de Puebla, México, 2016.
- [5] A. Soto Rodríguez, "E-Ciencias de la Información El plagio y su impacto a nivel académico y profesional," Jan. 2012.
- [6] Cisco Networking Academy, "Cisco Skills For All," Apr. 23, 2023. <https://skillsforall.com/es/launch?id=728abbf2-e134-4203-9365-b10e675e9971&tab=curriculum&view=62449c7c-a047-5c0f-8a08-640ed53efaac> (accessed May 30, 2023).

- [7] R. G. Duque, "Python PARA TODOS", Accessed: Jun. 18, 2023. [Online]. Available: <http://mundogeek.net/tutorial-python/>
- [8] "¿Qué es una red neuronal? - Documentación de IBM." <https://www.ibm.com/docs/es/spss-statistics/29.0.0?topic=networks-what-is-neural-network> (accessed Jun. 18, 2023).
- [9] X. B. Olabe, "REDES NEURONALES ARTIFICIALES Y SUS APLICACIONES".
- [10] P. Larrañaga, "Tema 8. Redes Neuronales".
- [11] Marcos, "Introducción a las Redes de Neuronas Artificiales", Accessed: Jun. 18, 2023. [Online]. Available: <http://sabia.tic.udc.es/>
- [12] "Conceptos básicos (redes neuronales) - Documentación de IBM." <https://www.ibm.com/docs/es/spss-modeler/saas?topic=networks-basics-neural> (accessed Jun. 18, 2023).
- [13] Antonio J. Serrano, Emilio Soria, and Jose D. Maritn, *REDES NEURONALES ARTIFICIALES*.
- [14] C. Alberto Ruiz Marta Susana Basualdo Autor and D. Jorge Match, "Cátedra: Informática Aplicada a la Ingeniería de Procesos-Orientación I Redes Neuronales: Conceptos Básicos y Aplicaciones".
- [15] I. Bonet Cruz, S. Salazar Martinez, A. R. Abed, G. Abalo, and M. M. G. Lorenzo, "Redes neuronales recurrentes para el análisis de secuencias Recurrent neural network for sequences analysis".
- [16] J. Pérez Guerrero Sevilla, J. de, T. por, and R. Pino Mejías, "REDES RECURRENTEES."
- [17] "Siamese Network : Architecture and Applications in Computer Vision Hengliang Luo," 2014. [Online]. Available: <http://en.wikipedia.org/wiki/Siamese>
- [18] L. Mera-Jiménez and J. F. Ochoa-Gómez, "Redes neuronales convolucionales para la clasificación de componentes independientes de rs-fMRI," *TecnoLógicas*, vol. 24, no. 50, p. e1626, Jan. 2021, doi: 10.22430/22565337.1626.
- [19] A. : Javier, S. Gozalo, and F. Díaz Gómez, "Análisis del estado del arte de la generación de texto con redes neuronales mediante modelos de Transformer."
- [20] G. García Subies Tutor and F. Serradilla García Madrid, "Trabajo Fin de Máster Modelos de Transformers para la Clasificación de Texto."
- [21] L. Laura, R. Calcagni, and F. Ronchetti, "Redes Generativas Antagónicas y sus aplicaciones."
- [22] J. de la Torre, "REDES GENERATIVAS ADVERSARIAS (GAN) FUNDAMENTOS TEÓRICOS Y APLICACIONES SURVEY," 2023.
- [23] G. Koch, R. Zemel, and R. Salakhutdinov, "Siamese Neural Networks for One-shot Image Recognition."
- [24] "Contenido Lista de figuras IV."
- [25] L. V Utkin, M. S. Kovalev, and E. M. Kasimov, "An explanation method for Siamese neural networks."
- [26] "Contenido Lista de figuras IV."
- [27] Mario Campos Mocholi, "Clasificación de textos basadas en redes neuronales".
- [28] K. Beck, "Praise for Extreme Programming Explained, Second Edition."

- [29] Kent. Beck and M. Fowler, *Planning extreme programming*. Addison-Wesley, 2001.
- [30] Y. Blanco, I. E. Blanco, C. Arletys, and I. L. Robles, "Metodologías de desarrollo de software (XP)."
- [31] Roger s. Pressman, *Ingeniería del software-Presman 7ma edición 2010*.
- [32] N. Matallana Torres and J. Antonio Torres Benavides PRESIDENTE Maria Ysabel Aranguri Garcia Manuel Gregorio Leon Tenorio, "Aplicación web utilizando la metodología de diagnóstico logístico para apoyar el proceso de gestión de pedidos en una universidad privada de la región Lambayeque PRESENTADA POR INGENIERO DE SISTEMAS Y COMPUTACIÓN APROBADA POR."
- [33] P. Letelier and M. C. Penadés, "Métodologías ágiles para el desarrollo de software: eXtreme Programming (XP)," *www.cyta.com.ar/ta0502/v5n2a1.htm*, Apr. 2006.
- [34] Luis Miguel Echeverry and Luz Elena Delgado, "CASO PRÁCTICO DE LA METODOLOGÍA ÁGIL XP AL DESARROLLO DE SOFTWARE," Universidad Tecnológica de Pereira , 2007.
- [35] F. Pedregosa *et al.*, "Scikit-learn: Machine learning in python Machine Learning for neuroimaging View project Scikit-learn: Machine Learning in Python Gaël Varoquaux Bertrand Thirion Vincent Dubourg Alexandre Passos PEDREGOSA, VAROQUAUX, GRAMFORT ET AL. Matthieu Perrot," 2011. [Online]. Available: <http://scikit-learn.sourceforge.net>.
- [36] Gavin. Hackeling, *Mastering Machine Learning with scikit-learn*. Packt Publishing, 2014.
- [37] Carlos Antona Cortes, "HERRAMIENTAS MODERNAS EN REDES NEURONALES: LA LIBRERÍA KERAS," 2017.
- [38] M. Scarpino, *TensorFlow For Dummies*. John Wiley & Sons, 2018.
- [39] A. A. Estévez Acosta and others, "Desarrollo e implementación de un sistema de identificación de signos de la lengua de señas mexicana basado en redes neuronales y el sistema embebido Jetson Nano," *REPOSITORIO NACIONAL CONACYT*, 2022.

11. Anexos

a. Anexo 1

Revisión de Trabajos Relacionados.

En el siguiente link se adjunta todos los documentos encontrados:

<https://drive.google.com/drive/folders/1cJEhcbCCawOzR2HWc0tR6l1OKDYvWsnr?usp=sharing>

b. Anexo 2

En el siguiente link se adjunta los Trabajos Relacionados seleccionados de toda la búsqueda realizada:

<https://drive.google.com/drive/folders/1aaTpeSEmAlpfTPhM4dA83do3VP0M4StB?usp=sharing>

c. Anexo 3

Búsqueda de Librerías para trabajar con redes neuronales

El siguiente link proporciona los documentos encontrados de la investigación de librerías:

<https://drive.google.com/drive/folders/1MGUD9Z6qAmAAQl0Rws9HgVimdMdqNh10?usp=sharing>

d. Anexo 4

Link del Dataset Generado:

<https://drive.google.com/drive/folders/1KreGlsbhM2UsjYQfyVg0ZL4x8EBMj0o6?usp=sharing>

e. Anexo 5

Código del detector de plagio generado:

<https://github.com/SantiagoRoman26/Plagio>

f. Anexo 6

Código de la red Neuronal Generada:

https://github.com/SantiagoRoman26/Plagio/blob/main/src/python/redes_neuronales.py