UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

Posgrado en Ciencia e Ingeniería de la Computación

Verificación de autoría de discurso cruzado basada en grafos heterogéneos

T E S I S

QUE PARA OPTAR POR EL GRADO DE:

MAESTRO EN CIENCIA E INGENIERÍA DE LA COMPUTACIÓN

P R E S E N T A:

MARTÍNEZ GALICIA JORGE ALFONSO TONATIUH

DIRECTOR DE TESIS:
DRA. HELENA M. GÓMEZ ADORNO
Instituto de Investigaciones en Matemáticas Aplicadas y Sistemas

C.U., Ciudad de México Noviembre, 2023

Jurado asignado:

Presidente:	Dr. Gibrán Fuentes Pineda
Vocal:	Dra. Helena Montserrat Gómez Adorno
Secretario:	Dra. Gemma Bel Enguix
Suplente:	Dr. José Antonio Neme Castillo
Suplente:	Dr. Iván Vladimir Meza Ruíz

Directora de tesis:

_____

Dra. Helena M. Gómez Adorno

Dedicatoria

Hoy, al finalizar este importante capítulo de mi vida académica, quiero dedicar estas palabras a ustedes, quienes han sido fundamentales en mi trayecto hacia el logro de esta tesis. Su presencia constante y su apoyo han sido el motor que me ha impulsado a alcanzar mis metas.

A mi familia, a mi esposa y a quienes siempre he considerado como mi pilar más sólido, quiero expresar mi gratitud. Su paciencia, comprensión y amor incondicional han sido mi fuerza inspiradora durante todo este camino. Desde el comienzo, me brindaron su apoyo emocional, me alentaron cuando dudaba de mí mismo y celebraron conmigo cada pequeño avance. Su respaldo inquebrantable me ha dado la confianza para superar obstáculos y perseverar hasta el final. Este logro no solo es mío, sino también de ustedes.

El logro de esta tesis es un testimonio de la fuerza colectiva que se genera cuando nos rodeamos de personas valiosas. Cada uno de ustedes ha dejado una huella imborrable en mi vida y, sin duda alguna, su influencia perdurará en mi carrera profesional y personal.

Agradecimientos

Deseo expresar mi más profundo agradecimiento a cada uno de ustedes. su guía, apoyo y contribuciones han sido fundamentales para hacer posible el logro de esta tesis.

Al Posgrado en Ciencias e Ingeniería de la Computación, al que pertenezco, quiero agradecer por brindarme la oportunidad de formarme en un entorno académico de excelencia. La dedicación a la educación han permitido que adquiera los conocimientos y las herramientas necesarias para llevar a cabo esta investigación. Agradezco por la calidad de la enseñanza y el enfoque en la investigación que me han proporcionado, lo cual ha sido invaluable para mi crecimiento profesional.

A mi tutor la Dra. Helena M. Gómez Adorno, agradezco sinceramente por su orientación, paciencia y sabiduría a lo largo de este proceso. Su experiencia y compromiso han sido cruciales para mi desarrollo como investigador. Gracias por desafiarme constantemente, por ayudarme a ampliar mis horizontes y por brindarme la confianza y autonomía necesarias para explorar nuevas ideas. Su apoyo incondicional me ha impulsado a superar mis propios límites y alcanzar resultados.

Hoy, celebro la culminación de esta tesis, pero también celebro la relación que hemos construido a lo largo de este proceso. Espero poder devolverles de alguna manera el apoyo que me han brindado, inspirando a otros, compartiendo conocimiento y siendo una fuerza positiva en la comunidad académica.

Índice general

Índice de figuras

Índice de tablas

Verificación de autoría de discurso cruzado basada en grafos heterogéneos
Tesis de Maestría

Jorge Alfonso Tonatiuh Martínez Galicia
Posgrado en Ciencia e Ingeniería de la Computación
Universidad Nacional Autónoma de México

Resumen

La presente tesis se centra en la tarea de verificación de autoría de discurso cruzado, un área de investigación que tiene como objetivo determinar si dos textos fueron escritos o no por un mismo autor con el propósito de identificar posibles plagios o fraudes académicos. En particular, se propone un modelo basado en grafos para abordar este desafío.

El modelo propuesto se basa en la representación de documentos como grafos, donde cada nodo representa una entidad textual, como una palabra o un texto completo, y las relaciones entre los nodos capturan la co-ocurrencia o la proximidad léxica en el texto. Estas relaciones son fundamentales para identificar los patrones de escritura únicos de cada autor.

La tesis se enfoca en la construcción de un grafo heterogéneo, utilizando técnicas de procesamiento de lenguaje natural y análisis de grafos. Se implementa un modelo para la extracción de características relevantes de los documentos y la construcción de los grafos correspondientes. Estas características incluyen las categorías gramaticales, la frecuencia de término y otros atributos lingüísticos.

Posteriormente, se emplea un enfoque de aprendizaje automático para entrenar un modelo de clasificación capaz de distinguir entre documentos escritos por diferentes autores. Se utilizan redes convolucionales para capturar patrones y características distintivas en los grafos. El modelo se entrena utilizando el conjuntos de datos del PAN2022 [28], donde se conocen las autorías de los documentos.

Para evaluar la efectividad del modelo propuesto, se realizan experimentos utilizando diferentes valores de hiperparámetros y métricas de evaluación propuestas en [28]. Los resultados muestran que el modelo basado en grafos logra un buen desempeño en la verificación de autoría de discurso cruzado, superando a otros enfoques tradicionales basados en características lingüísticas o estadísticas.

En resumen, esta tesis presenta un enfoque innovador para abordar la tarea de verificación de autoría de discurso cruzado mediante un modelo basado en grafos. El modelo propuesto demuestra su capacidad para identificar patrones de escritura únicos y distinguir entre diferentes autores sin importar el tipo de discurso con un rendimiento aceptable.

Capítulo 1

Introducción

El análisis de autoría es una línea de investigación en el procesamiento de lenguaje natural (PLN), tiene una larga historia, que incluyen algunos casos famosos de autoría en disputa, así como varias aplicaciones forenses. Dentro del análisis de autoría existe un problema fundamental que es la atribución de autoría, el cual busca identificar el autor de un texto dentro de un conjunto de autores. A este tipo de problema de atribución se le llama de conjunto cerrado, debido a que se conocen los autores, y el otro caso es de conjunto abierto, es decir, no se conoce el autor y no es necesario saberlo, en este caso especial es en donde entra la verificación de autoría. La tarea de verificación de autoría trata de identificar si dos textos fueron escritos o no por un mismo autor, sin importar la identidad del autor.

La representación del texto a ser analizado, es el primer reto en esta tarea, hoy en día toda la información es producida y encontrada en distintos formatos como son revistas, libros, artículos científicos, etc. Esto nos lleva a proponer una efectiva representación del texto. En enfoques tradicionales se extraen características del texto, los documentos también pueden ser representados con palabras o n-gramas usando vectores de características dentro del marco del modelo de espacio vectorial (VSM) y se usan para entrenar algoritmos de clasificación supervisados o en la similitud. Estos métodos ignoran la semántica y sintaxis de los documentos al representar el texto como una bolsa de palabras y al ser entrenados y evaluados utilizando textos de diferente tipo de discurso, como pueden ser correos electrónicos, mensajes de texto, ensayos, etc. el rendimiento de estos disminuye [20].

En PLN, los grafos son usados para representar textos en etapas de pre-procesamiento, como vinculación textual, desambiguación de sentido de palabra, etc. [78]. Los grafos proveen información adicional contenida en los textos que no se puede obtener cuando se procesan de la manera secuencial tradicional. Así como se busca aplicar maneras mas eficientes de representar el texto, también se busca encontrar un modelo que mejor convenga para este tipo de estructuras de datos, como lo presentan en [30] donde utilizan una red siamesa basada en grafos para resolver la tarea de verificación de autoría, obteniendo muy buenos resultados cuándo los textos son del mismo tipo de discurso.

En este documento se presenta una propuesta de un enfoque de verificación de autoría utilizando representación en grafos para los textos de un corpus de distinto tipo de discurso de diferentes autores [28], aplicando un modelo de redes convolucionales sobre grafos para clasificación de texto presentado en [115].

1.1. Motivación

Con el aumento de la tecnología de la información y la comunicación, la cantidad de textos que se generan día con día en internet es inmensa, es por eso que, campos de estudio como lo es el procesamiento de lenguaje natural (PLN) han obtenido gran relevancia. En PLN específicamente en el análisis de autoría existe un área de investigación que es la atribución de autoría la cual trata de revelar la identidad de sus autores por medio del estilo de escritura.

La verificación de autoría es un problema particular de la atribución de autoría de conjunto abierto, es decir, no es necesario conocer al autor, simplemente es dados dos textos decidir si estos dos fueron escritos o no por el mismo autor. Esta es una tarea importante en muchas aplicaciones, como la investigación forense, disputas de autoría y la protección contra la falsificación.

Existen diversos estudios relacionados a la verificación de autoría, pero en particular un caso que aún requiere de investigación es el de verificación de autoría de discurso cruzado, en el que los textos a ser analizados no son necesariamente del mismo tipo de discurso, es decir, que uno de ellos puede ser un ensayo mientras que el otro un correo electrónico. Por lo tanto, es necesario desarrollar técnicas eficaces para verificar la autoría en este tipo de casos.

El objetivo de esta tesis es investigar y desarrollar un nuevo método basado en grafos para la verificación de autoría de discurso cruzado que sea preciso y eficiente. Al lograr esto, se pueden mejorar los procesos de verificación de autoría en muchas aplicaciones importantes.

En resumen, la motivación para esta tesis es la importancia creciente de la verificación de autoría de discurso cruzado en un mundo cada vez más digital y la necesidad de desarrollar técnicas más efectivas para resolver disputas de autoría y garantizar la autenticidad.

1.2. Objetivos

El objetivo general de este trabajo consiste en:

> Analizar, proponer e implementar una representación de texto basada en grafos que favorezca la identificación de los estilos de escritura de diferentes autores en distintos tipos de discurso, así como, desarrollar un modelo de red neuronal que utilice esta representación de texto y modele los documentos completos aplicado a la verificación de autoría de discurso cruzado.

Los objetivos particulares consisten en:

- Buscar y seleccionar un conjunto de datos que cumpla las características requeridas para la tarea de verificación de autoría de discurso cruzado.
- Realizar la limpieza, así como, análisis de los meta-datos que puedan ayudar al modelo.
- Entrenar y evaluar los modelos tradicionales junto con el propuesto en [30] utilizando el corpus.
- Proponer e implementar un método para transformar el texto a grafo.
- Implementar un modelo de red neuronal que pueda generar representaciones de documentos completos.
- Entrenar y evaluar el modelo utilizando el corpus seleccionado.
- Demostrar mejoría en las métricas de evaluación propuestas en [28], en comparación a los métodos tradicionales y al propuesto en [30].

1.3. Hipótesis

Mediante la aplicación de técnicas de procesamiento de lenguaje natural y de aprendizaje profundo, como el análisis de características lingüísticas, representación de textos como grafos y redes neuronales, permitirá identificar patrones de escritura únicos para cada autor y lograr una mejora en precisión en comparación a los métodos tradicionales en la verificación de autoría de discurso cruzado.

1.4. Preguntas de investigación

Las preguntas de investigación que surgen en este trabajo son:

1. ¿Es posible determinar si dos textos de distinto tipo de discurso fueron escritos por el mismo autor?
2. ¿Cuáles son las características lingüísticas y estilísticas más relevantes para la verificación de autoría en diferentes tipos de discurso, como correos electrónicos, mensajes de texto, ensayos y memos?
3. ¿Cuáles son las técnicas computacionales y modelos base más utilizados en la verificación de autoría de discurso cruzado, y cómo se comparan en términos de las métricas de evaluación propuestas en la tarea de verificación de autoría de discurso cruzado del PAN [28]?

1.5. Contribución

Las contribuciones de este trabajo son:

1. Revisión conceptual y estudio de la representación de textos como grafos para la verificación de autoría de discurso cruzado.
2. Definición de aspectos importantes a considerar en los corpus para verificación de autoría.
3. Modelo de red neuronal basada en grafos que modele documentos completos, aplicado al análisis de autoría, en específico, verificación de autoría de discurso cruzado.

1.6. Organización de la tesis

A continuación se describen los capítulos que componen esta tesis. Primero en el Capítulo 2 llamado *Antecedentes* se presenta una seria de conceptos y términos que se utilizarán en el desarrollo de esta tesis, necesarios para entender el proceso y desarrollo de la metodología así como los resultados. Posterior a esto, se aborda el Capítulo 3 *Verificación de Autoría*, en el cuál se presenta la definición formal de la tarea de verificación de autoría de discurso cruzado, así como un panorama global del estado del arte en el área de análisis de autoría, en específico en la tarea definida. Teniendo los conocimientos necesarios para entender este trabajo, se presenta el Capítulo 4 *Metodología grafo de texto heterogéneo*. En este capítulo, se describe la arquitectura del modelo propuesto para resolver la tarea de verificación de autoría de discurso cruzado, utilizando las propiedades de los grafos para representar textos. El Capítulo5 *Resultados y configuración experimental*, se describen los pasos realizados para el pre-procesamiento

del corpus, se muestran los resultados de las evaluaciones realizadas con el modelo propuesto variando distintos hiperparámetros. También se describen los modelos tradicionales y del estado del arte utilizados para comparar los resultados obtenidos con el modelo propuesto. Finalmente, el Capítulo 6 *Conclusiones y trabajo futuro*, describe los hallazgos encontrados en la experimentación y resultados que ayudan a un modelo de este tipo a solucionar la tarea de verificación de autoría de discurso cruzado, es decir, las implicaciones que tienen los resultados sobre las preguntas de investigación. También se proponen experimentos y cambios que podrían dar una continuidad a la mejora del modelo.

Capítulo 2

Antecedentes

En este capítulo se describen conceptos y términos que se utilizarán en el desarrollo de esta tesis, relacionados con el procesamiento del lenguaje natural, atribución de autoría, verificación de autoría, verificación de autoría de discurso cruzado, representación de características de textos y redes neuronales. En el apartado de representación de características y textos, se abordan temas como medidas de relación entre palabras y documentos, n-gramas, vectores de palabras, vectores de documentos, etc. En los temas relacionados a redes neuronales, se expone un resumen de los distintos métodos base utilizados para resolver la tarea de verificación de autoría. Estos conceptos son necesarios para entender capítulos posteriores en el documento.

2.1. Procesamiento de lenguaje natural

El lenguaje natural es el lenguaje hablado o escrito por los seres humanos para propósitos de comunicación. El procesamiento de lenguaje natural o PLN es una rama de la inteligencia artificial y la lingüística computacional que busca desarrollar algoritmos y modelos que permitan entender, interpretar y generar lenguaje humano de manera precisa, lo que puede tener múltiples aplicaciones en la clasificación de textos, análisis de sentimientos, análisis de autoría, traducción automática, entre otros.

Lo que distingue las aplicaciones de procesamiento de lenguaje de otras aplicaciones de procesamiento de datos es su uso de conocimiento del lenguaje, un ejemplo de esta comparativa que se expone en [53] es el programa **wc** en Unix, el cual es usado para contar el número total de bytes, palabras y líneas en un archivo de texto. Cuando se usa para contar bytes líneas, **wc** es una aplicación de datos ordinaria. Sin embargo, cuando es usado para contar palabras en un archivo requiere conocimiento sobre que es una palabra, y por lo tanto se convierte en un sistema de procesamiento de lenguaje.

Claro que, el programa **wc** es un sistema muy simple con un conocimiento del lenguaje muy limitado. Pero existen agentes conversacionales muy sofisticados como ChatGPT [86] los cuales requieren de un conocimiento del lenguaje mucho mas amplio. Para resumir, involucrarse en un comportamiento lingüístico complejo requiere varios tipos de conocimiento del lenguaje:

- Discurso: conocimiento sobre unidades lingüísticas más grandes que un solo enunciado.
- Pragmática: conocimiento de la relación del significado con el objetivo y las intenciones del hablante.
- Semántica: conocimiento del significado.

- Sintaxis: conocimiento de las relaciones estructurales entre las palabras.
- Morfología: conocimiento de los componentes significativos de las palabras.
- Fonética y Fonología: conocimiento sobre los sonidos lingüísticos.

2.2. Análisis de autoría

El análisis de autoría se enfoca en la identificación del estilo de escritura del autor y en la comparación de ese estilo con otros textos del mismo autor o con textos de otros autores para identificar similitudes y diferencias [47]. El objetivo principal del análisis de autoría es comprender el estilo de escritura de un autor y cómo éste se manifiesta en diferentes textos. El análisis de autoría se basa en técnicas de análisis lingüístico y estadístico para identificar patrones de uso de palabras, frases, y otros elementos lingüísticos que pueden ser características distintivas de un autor.

El análisis de autoría supone que cuando un autor escribe un texto usa ciertas palabras y combinaciones de ellas, ya sea inconsciente o conscientemente, con lo que surgen patrones específicos que caracterizan el estilo de escritura de un autor y hacen que su escritura sea única de tal manera que es posible extraer características del texto que sean capaces de diferenciar a los autores mediante el uso de técnicas estilísticas o de aprendizaje automático [47]. Estas técnicas se aplican para resolver distintos problemas, como la autoría de un texto en disputa, aplicaciones forenses, plagio, etc. La tareas que derivan del análisis de autoría para dar respuesta a este tipo de problemas es la atribución de autoría.

2.2.1. Atribución de autoría

La atribución de autoría se centra en la determinación de la autoría de un texto desconocido o en disputa y se basa en técnicas de análisis de autoría. Podemos definir este problema de la siguiente manera: si tenemos un conjunto de textos con autoría conocida, entonces es posible determinar el autor de un nuevo documento no visto.

Bajo esta definición, se pueden distinguir dos enfoques principales [48].

- Conjunto cerrado: Se basa en que dados una muestra de textos de autoría conocida y un texto del cual se desconoce su autoría, se puede asumir que el autor real de este texto es uno de los candidatos especificados de los que se dispone en las muestras de textos.
- Conjunto abierto: El autor real podría ser uno de los propuestos autores u otro desconocido. Este tipo de casos requiere de un análisis de atribución mas complejo debido a que es posible obtener una respuesta referente a que no es posible atribuir la autoría debido a falta de evidencia o prueba fallida. Dicho esto, existen casos especiales en el atribución de autoría en donde no es necesario identificar al autor, simplemente verificar si un texto de autoría desconocida comparte características estilísticas a un texto de autoría conocida. Esta tarea es conocida como verificación de autoría.

2.2.2. Verificación de autoría

La verificación de autoría es un caso especial de la atribución de autoría de conjunto abierto en el que no es necesario conocer al autor, la tarea de verificación de autoría provee una respuesta binara y el objetivo es, dada una pareja de textos (normalmente uno de ellos de autoría conocida y el otro de autoría desconocida) determinar si los documentos fueron o

no escritos por el mismo autor [48]. Con lo que, este caso se puede abordar como un problema de clasificación de una sola clase [61].

La efectividad de los modelos de verificación de autoría dependen de diferentes factores. Naturalmente, cuando la longitud de los textos proporcionados es muy corta suele deteriorar la respuesta de los modelos. Otros factores que complican a esta tarea son cuando los textos pertenecen a distintos temas (política vs. deporte), a distintos géneros (ensayos vs. crítica), a distinto lenguaje (Inglés vs. Español) o tipo de discurso (correos electrónicos, ensayos, mensajes de texto, etc).

En la verificación de autoría de discurso cruzado los textos proporcionados pertenecen a distintos tipos de discurso. Los tipos de discurso tienen diferencias significativas en términos de propósito comunicativo, audiencia, o formalidad. Por ejemplo, los tipos discursivos de los ensayos y los mensajes de texto enviados a familiares tienen importantes diferencias estilísticas. Por lo tanto, es muy difícil distinguir las características del autor que permanecen intactas en todos los tipos de discurso [28].

2.3. Representación de características de los textos

Como ya se mencionó anteriormente el objetivo principal del análisis de autoría es comprender el estilo de escritura de un autor y cómo éste se manifiesta en diferentes textos apoyado de herramientas computacionales. Es por ello que para iniciar este proceso, es necesario tener representaciones de las características que definen a los estilos de escritura. Estas representaciones pueden ser obtenidas mediante, caracteres, palabras, relación entre palabras, relación entre palabras y textos, categorías gramaticales de las palabras o etiquetas POS por sus siglas en inglés.

Las técnicas de verificación de autoría se basan en algoritmos y modelos computacionales que requieren una representación estructurada y cuantificable de los textos. Estas representaciones permiten a los algoritmos analizar y comparar características lingüísticas y estilísticas en los textos para identificar patrones y señales distintivas de la autoría [97]. Además, la representación de los textos en un formato comprensible para las computadoras facilita el procesamiento automático y la extracción de características relevantes. Al utilizar técnicas como el análisis sintáctico, la extracción de entidades y la modelización de lenguaje, la computadora puede identificar elementos clave en el texto, como palabras, frases, estructuras gramaticales y relaciones semánticas [3].

2.3.1. Modelo de espacio vectorial

El modelo de espacio vectorial es un enfoque muy utilizado en PLN para representar y procesar el significado de los términos o palabras en un conjunto de textos. Cada documento es representado con un vector en un espacio de n-dimensiones, donde cada dimensión corresponde a un término en el conjunto de textos [80]. Este modelo es muy útil en tareas de recuperación de información, agrupación y clasificación de documentos, extracción de palabras clave y similitud entre textos [85]. Además, el modelo es escalable y puede ser mejorado utilizando técnicas de aprendizaje automático, representación de documentos y palabras más avanzadas, modelos de redes neuronales, que no solo capturan la relación entre palabras cercanas, sino también, pueden aprender la semántica y el contexto de un documento completo.

Existen distintas técnicas para llevar acabo la construcción de un modelo de espacio vectorial, como el uso de bolsa de palabras, frecuencia de término, frecuencia de término – frecuencia inversa de documento, n-gramas, etc. Generalmente se siguen los siguientes pasos [53]:

- Preprocesamiento de datos: se eliminan las palabras que no aporten información al modelo, se eliminan las palabras poco frecuentes, etc.
- Construcción del vocabulario: se genera una lista de los tokens o términos que pueden ser palabras o n-gramas que aparecen en el corpus o conjunto de textos.
- Representación de los documentos: se representan los documentos como vectores en un espacio de n-dimensiones.

2.3.1.1. N-gramas

Los n-gramas son secuencias de n elementos de un texto. Por lo general, estos elementos son palabras, aunque también pueden ser caracteres, símbolos o cualquier otro tipo de unidad lingüística. Son útiles en PLN porque permiten capturar patrones y características del lenguaje, como la co-ocurrencia de palabras o la estructura sintáctica [53, 72]. Se utilizan comúnmente en tareas como el modelo del lenguaje [72], la clasificación de texto, la extracción de información y la traducción automática [53].

Por ejemplo, considerando el siguiente texto: “I am a Second year university student"
Los n-gramas serían las siguientes secuencias de palabras:

- 1-gramas o términos: “I”, “am”, “a”, “Second”, “year”, “university”, “student”.
- 2-gramas o bigramas: “I am”, “am a”, “a Second”, “Second year”, “year university”, “university student”.
- 3-gramas o trigramas: “I am a”, “am a Second”, “a Second year”, “Second year university”, “year university student”.

Ya obtenido este preprocesamiento del texto original, se puede construir una estructura de frecuencia de n-gramas. Tomando como ejemplo los bigramas quedaría de la siguiente forma:

{(“I am”):1,(“am a”):1,(“a Second”):1,(“Second year”):1,(“year university”):1,(“university student”):1}

En PLN, se pueden utilizar los n-gramas y su frecuencia para predecir la probabilidad de una palabra dada una secuencia de palabras anteriores en un texto. Los modelos de lenguaje basados en n-gramas son muy efectivos para muchos idiomas y son relativamente fáciles de construir y entrenar [53].

2.3.1.2. Bolsa de palabras

En PLN, una bolsa de palabras o “bag of words” en inglés, es una representación simplificada de un texto que consiste en una lista de las palabras que aparecen en él y su frecuencia. En esta representación, el orden de las palabras no se tiene en cuenta, es decir, se pierde la información de la estructura y la gramática del texto [53]. Suponiendo que alguien toma todas las palabras de una oración, las reordena de manera aleatoria y posteriormente obtiene la frecuencia de cada una de estas palabras en la oración. Estos son los pasos a seguir para obtener una bolsa de palabras por medio de una oración.

Por ejemplo, considerando el siguiente texto: “I am a Second year university student"
La bolsa de palabras sería la siguiente:

{“I”: 1, “am”: 1, “a”: 1, “Second”: 1, “year”: 1, “university”: 1, “student”: 1}

Esta técnica es comúnmente utilizada en la etapa de preprocesamiento en tareas de PLN como la clasificación de texto, recuperación de información, minería de textos, etc. Esta técnica puede ser mejorada eliminando palabras (stopwords) que no aporten información al modelo que se vaya implementar, normalización palabras a su forma base (lemmatización o stemming).

2.3.2. Vectores de palabras

Los vectores de palabras (también llamados word embeddings) son representaciones numéricas densas que capturan características semánticas y sintácticas de las palabras, permitiendo realizar cálculos y análisis basados en similitud y relaciones entre ellas. En lugar de representar las palabras como cadenas de texto, los vectores de palabras asignan a cada palabra un vector de n-dimensiones en un espacio vectorial [53]. Las palabras que aparecen en contextos similares, es decir, en oraciones, párrafos o textos similares, deben tener vectores de palabras similares.

Se puede utilizar el enfoque bolsa de palabras junto con técnicas de conteo y ponderación para obtener una representación vectorial de palabras en un contexto específico.

Un ejemplo sería el siguiente, suponiendo que se tiene un conjunto de documentos que consiste en las siguientes oraciones:

- El gato es negro.
- El perro es marrón.
- El cielo es azul.

Los pasos a seguir son los siguientes:

- Paso 1: Construcción del vocabulario. Se recopilan las palabras únicas de los documentos y se construye un vocabulario. En este caso, el vocabulario sería: [.El", "gato", .es", "negro", "perro", "marrón", çielo", .azul"].
- Paso 2: Representación de documentos en un espacio vectorial. Por cada documento, se crea un vector donde cada posición representa una palabra del vocabulario y su valor indica la frecuencia de esa palabra en el documento. Tomando como ejemplo la primera oración .El gato es negro.", su representación vectorial sería: [1, 1, 1, 1, 0, 0, 0, 0]. El valor 1 indica que las palabras .El", "gato", .esz "negro.están presentes en el documento, mientras que las demás palabras tienen un valor de 0 porque no aparecen en el documento.

De esta manera, para cada oración se tendría una representación vectorial correspondiente.

Es importante mencionar que la representación de bolsa de palabras no captura la semántica ni las relaciones entre las palabras, ya que solo cuenta la frecuencia de ocurrencia de las palabras en cada documento. Además, no considera el orden de las palabras en el texto. Para abordar estas limitaciones, se utilizan técnicas más avanzadas, como los word embeddings, como Word2Vec o GloVe, que generan vectores densos y capturan mejor el significado y las relaciones entre las palabras [48, 80]. Los vectores de palabras son una herramienta importante en PLN y se utilizan en una variedad de aplicaciones, como la clasificación de texto, la extracción de información y la traducción automática.

2.3.3. Vectores de documentos

Los vectores de documentos son representaciones numéricas de documentos de texto en un corpus. A diferencia de los vectores de palabras, que representan palabras individuales, los vectores de documentos representan documentos completos como vectores de n-dimensiones.

Los vectores de documentos se crean a partir de vectores de palabras y se utilizan para capturar información semántica sobre los documentos en un corpus. Existen varios métodos para crear vectores de documentos, como el modelo de espacio vectorial (VSM) y la reducción de dimensionalidad. [53, 66]

Los vectores de documentos tienen varias aplicaciones en PLN, como en la recuperación de información, la clasificación de documentos y la agrupación de documentos.

2.3.4. Grafos de texto

Un enfoque común y estándar para representar documentos de texto en PLN es bolsa de palabras. Este modelo es adecuado para capturar la frecuencia de las palabras. Sin embargo información estructural y sintáctica es ignorada. La representación basada en grafos es una construcción matemática que puede modelar información estructural y de relación entre palabras de una forma muy eficiente [95].

Los grafos de texto son una representación gráfica de las relaciones sintácticas entre las palabras en un texto o un conjunto de textos. En los grafos de texto, los nodos representan palabras y las aristas representan relaciones entre ellas. La representación basada en grafos proporciona cálculos relacionado con varias operaciones como el peso del término, la clasificación que es útil en muchas aplicaciones en la recuperación de información [20]. Hay varios tipos de relaciones que pueden ser representadas en un grafo de texto, como la coocurrencia de palabras en un mismo contexto, la relación de antónimos y sinónimos, la relación de causa y efecto, entre otros.

Existen diferentes representación de textos como grafos que se utilizan en PLN para capturar las relaciones semánticas y sintácticas entre las palabras. A continuación, se presentan algunas de estas arquitecturas:

- Gráfico sintáctico integrado (GSI): Este modelo es capaz de capturar la mayoría de las características disponibles en el texto, desde los niveles morfológicos hasta los semánticos y discursivos. Al incluir relaciones léxicas, sintácticas, morfológicas y semánticas en la representación, el modelo es capaz de integrar en el grafo diversas unidades de texto, como palabras, frases, cláusulas, oraciones, etc [35]. El GSI se construye a partir de los siguientes niveles del lenguaje:
  - Nivel léxico: El GSI puede representar elementos léxicos, como palabras, sin hacer referencia a la oración en la que aparecen [35].
  - Nivel morfológico: En este nivel, el GSI se ocupa de la identificación, análisis y descripción de la estructura de los morfemas del lenguaje, además de otras unidades lingüísticas como raíces, tallos, afijos y etiquetas de partes del discurso (POS) [35].
  - Nivel sintáctico: En este nivel, el GSI almacena reglas y principios que gobiernan la estructura de las oraciones, utilizando el formalismo de dependencias [35].
  - Nivel semántico: En este nivel, el GSI introduce el significado de una oración o texto en el grafo, centrándose en relaciones semánticas paradigmáticas como antonimia, sinonimia, inclusión de clase, parte-todo y caso [35].

La construcción del GSI implica analizar todas las oraciones del texto. Se obtienen los árboles de análisis sintáctico de las oraciones. Luego, se identifican los nodos de todos los árboles con la misma etiqueta, como los nodos correspondientes a diferentes ocurrencias del mismo tipo de palabra (lema y etiquetas POS), de modo que por cada tipo de palabra haya solo un nodo en el GSI. Dado que el árbol sintáctico de cada oración contiene el nodo raíz y todos los nodos raíz se colapsan en un solo nodo del GSI, el grafo resultante es conexo [35].

- Grafos semánticos: Los grafos semánticos representan las relaciones de significado entre las palabras. Cada nodo del grafo representa un concepto o una palabra, y los enlaces entre los nodos representan las relaciones semánticas, como sinónimos, antónimos o hiperónimos-hipónimos. Estos grafos permiten capturar el conocimiento semántico y son útiles para tareas como la desambiguación del sentido de las palabras y la respuesta a preguntas basada en el conocimiento [84].
- Grafos de dependencia: Los grafos de dependencia representan las relaciones gramaticales entre las palabras en una oración. Cada palabra se representa como un nodo del grafo, y las relaciones de dependencia, como el sujeto, el objeto y los modificadores, se representan mediante enlaces dirigidos entre los nodos correspondientes. Estos grafos capturan la estructura sintáctica de las oraciones y son útiles para tareas como análisis de sentimientos, resumen automático y respuesta a preguntas [72].
- Grafo de de co-ocurrencia: Los grafos de co-ocurrencia representan las relaciones de co-ocurrencia entre las palabras en un corpus de texto. Cada palabra se representa como un nodo, y los enlaces entre los nodos se basan en la frecuencia o la proximidad de co-ocurrencia de las palabras en el texto. Estos grafos ayudan a capturar las relaciones contextuales entre las palabras y se utilizan en técnicas como el algoritmo PageRank para el análisis de texto [30, 79].

  Los pasos para construir un grafo de co-ocurrencia son los siguientes:

  - Recopilación de datos: Se necesita un conjunto de textos o documentos sobre los cuales se construirá el grafo de co-ocurrencia. Estos textos pueden ser un corpus específico, un conjunto de documentos relacionados o cualquier otro conjunto de datos de texto relevante.
  - Preprocesamiento del texto: Se debe realizar un preprocesamiento del texto para limpiar y normalizar los datos. Esto puede incluir la eliminación de signos de puntuación, el manejo de mayúsculas y minúsculas, la eliminación de palabras irrelevantes (stop words) y la lematización o el stemming para reducir las palabras a su forma base.
  - Construcción de la matriz de co-ocurrencia: A partir de los textos preprocesados, se construye una matriz de co-ocurrencia que registra las frecuencias de aparición conjunta de las palabras en el corpus. Cada fila y columna de la matriz representa una palabra del vocabulario y los valores de la matriz indican la cantidad de veces que dos palabras co-ocurren en el mismo contexto.
  - Definición de la ventana de contexto: La ventana de contexto define la proximidad que se considera para determinar si dos palabras co-ocurren. Puede ser una ventana de palabras adyacentes o una ventana más amplia que abarque varias oraciones o párrafos. La elección de la ventana de contexto depende de la naturaleza de los textos y de la tarea específica.

- Construcción del grafo de co-ocurrencia: Utilizando la matriz de co-ocurrencia, se construye el grafo de co-ocurrencia donde cada palabra representa un nodo y los enlaces entre los nodos representan la fuerza de la co-ocurrencia entre las palabras. La fuerza de la co-ocurrencia se puede medir mediante medidas como la frecuencia de co-ocurrencia, la similitud del coseno o la información mutua.

En la Figura 2.1 se muestra un ejemplo de un grafo de co-ocurrencia para la oración "Momo, also known as The Grey Gentlemen or The Men in Grey", en el cuál se agrega el token y su categoría gramatical en cada uno de los nodos.

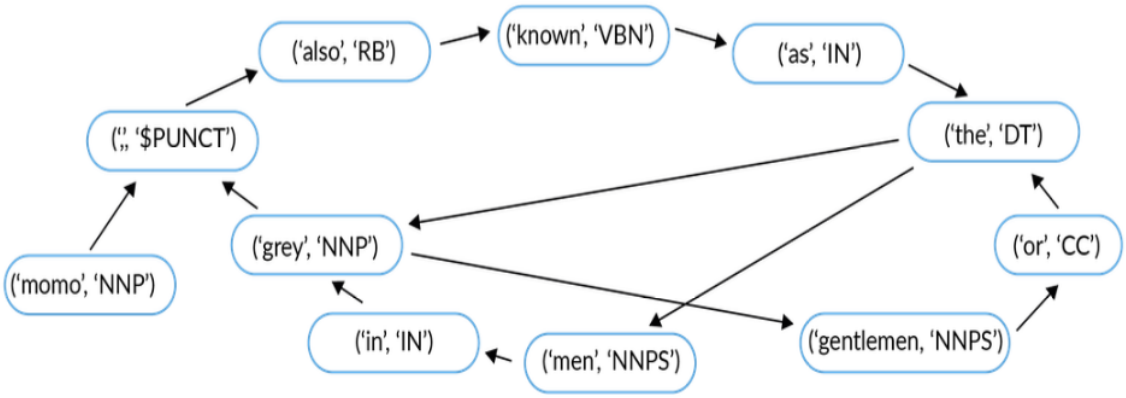

Figura 2.1: Grafo de co-ocurrencia.

Es importante tener en cuenta que la construcción de un grafo puede variar según el enfoque y las necesidades específicas de cada proyecto. La representación de los grafos de texto puede ser utilizada en combinación con otros métodos de PLN, como la modelización de temas y la clasificación de textos, para mejorar el rendimiento de los modelos [115].

2.4. Esquemas de pesado

En PLN, los esquemas de pesado son técnicas utilizadas para asignar pesos a las palabras en un conjunto de textos o corpus. La idea en la que se basa esta técnica es en que algunas palabras pueden ser más relevantes que otras en un documento o corpus y, por lo tanto, deberían tener un peso mayor para el análisis [53, 72].

Existen varios esquemas de pesado para el análisis de textos, como la frecuencia de términos, la frecuencia inversa de documentos (TF-IDF), y la información mutua puntual (PMI). Cada esquema de pesado tiene sus propias fortalezas y debilidades, y se utilizan para diferentes tipos de tareas de análisis de texto [72].

2.4.1. TF-IDF

TF-IDF es una técnica utilizada en PLN para evaluar la importancia de una palabra en un documento, en relación con un conjunto de documentos. Está compuesta por dos medidas para obtener un valor que indica que tan relevante es una palabra para un documento en particular:

- Frecuencia de término o TF por sus siglas en inglés, mide la frecuencia de una palabra en un documento, es decir, cuántas veces aparece la palabra en el documento.

- Frecuencia inversa de documento o IDF por sus siglas en inglés, mide que tan relevante es una palabra en un conjunto de documentos, considerando cuántos documentos contienen dicha palabra [51, 53].

Definido lo anterior, las palabras que aparecen con frecuencia en un documento pero no en otros documentos tendrán un valor TF-IDF alto, esto indica que estas palabras son importantes para un documento en específico. Esta medida está definida por la siguiente ecuación:

$$TF - IDF = TF \times log(\frac{N}{DF}) \tag{2.1}$$

Donde, N es el número total de documentos en el conjunto de textos y DF el número de documentos que contienen la palabra.

2.4.2. PMI

La información mutua puntual o PMI por sus siglas en inglés, es una medida estadística utiliza en PLN para medir la asociación entre dos palabras. Esta se basa en la probabilidad conjunta de que dos palabras aparezcan juntas en un conjunto de textos o corpus y la probabilidad individual de cada palabra. El PMI tiene un valor mayor para una pareja de palabras que aparecen juntas con mayor frecuencia [53]. Esta medida es utilizada a menudo en la identificación de términos importantes en un corpus, la generación de palabras clave y la detección de relaciones semánticas entre palabras [48].

El PMI de un par de palabras se puede calcular con la siguiente ecuación:

$$PMI(w1, w2) = log(\frac{P(w1, w2)}{P(w1) * P(w2)}) \tag{2.2}$$

Donde, P(w1, w2) es la probabilidad conjunta de que w1 y w2 aparezcan juntas en un corpus. P(w1) y P(w2) son las probabilidades individuales de w1 y w2 en el corpus.

2.5. Redes neuronales

Las redes neuronales están inspiradas en la estructura y el funcionamiento del cerebro humano, están diseñadas para reconocer patrones complejos y aprender a partir de ejemplos. Recientemente se han propuesto diferentes modelos de redes neuronales para resolver tareas difíciles como el reconocimiento de imágenes. En PLN, estos tipos de modelos se han utilizado exitosamente, por ejemplo, en autoría o reconocimiento de discurso, análisis de sentimientos, o traducción automática [36].

En las redes neuronales, el bloque de construcción fundamental es una neurona artificial que corresponde a una simple unidad de procesamiento. Como se describe en la Figura 2.2, una neurona está compuesta de un estructura interna y tiene conexiones con otras neuronas. En la Figura 2.2, se podría interpretar que has conexiones con 5 neuronas en el lado izquierdo las cuales pertenecen a una capa previa y dos neuronas a la siguiente capa del lado derecho [48].

El trabajo realizado por la neurona artificial comprende tres pasos:

- En el primer paso, la neurona computa una suma pesada de todas las entradas de la capa previa. Formalmente se puede definir esta suma como net_j 2.3 para la jth neurona teniendo m conexiones con la capa previa.

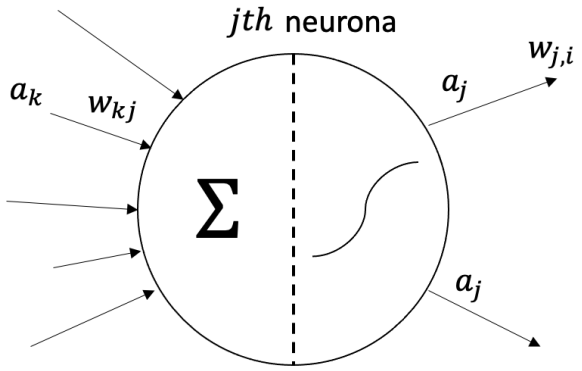

Figura 2.2: Representación de una sola neurona [48].

$$net_j = \sum_{k=1}^{m} w_{k,j} \cdot a_k + b_j \tag{2.3}$$

Donde $w_{k,j}$ es el peso asociado con la conexión entre la neurona k y j, y a_k indica el valor de activación transmitido por la neurona kth. Cuando el peso entre dos neuronas es positivo, ambas neuronas tienden a ser activadas al mismo tiempo. Con un peso negativo, la primer neurona inhibe a la segunda. En la Figura 2.2, esta función esta simbolizada por Σ dentro de la neurona.

- Para el segundo paso, la neurona determina el valor de activación que será enviado al siguiente nivel. Este valor es definido a través de una función de activación presentada en el lado derecho dentro de la neurona de la Figura 2.2. Como función de activación, se puede utilizar la *sigmoid*, *tanh*, *step*, entre otras. En este caso, si la función *tanh* es aplicada, el valor de activación denotado como a_j esta definido entre -1 y 1. Como se observa en la Figura 2.3.

- En el tercer paso, el valor de activación alcanzado a_j es propagado a las neuronas conectadas pertenecientes a la siguiente capa. Y la misma secuencia computacional es ejecutada para las siguientes capas.

Con tal bloque definido, una red neuronal puede ser organizada en capas. Con esta perspectiva, una red neuronal clásica llamada arquitectura multi-capa está compuesta de tres capas llamadas, capa de entrada, capa oculta y capa de salida [48].

En la Figura 2.4 se muestra un ejemplo de este tipo de arquitectura, esta arquitectura ha sido propuesta en [76] para resolver problemas de atribución de autoría entre Shakespeare y Fletcher. En esta arquitectura se utiliza una operación $Softmax$ como función de activación, la cuál transforma un vector de valores en una distribución probabilística.

Existen distintos tipos y arquitecturas de redes neuronales, las cuales se deben de aplicar dependiendo el problema o tarea que se requiera solucionar. Un tipo de red neuronal muy utilizada en el área de análisis de autoría y son las redes neuronales convolucionales, que se pueden utilizar tanto para análisis de imágenes como para análisis de texto.

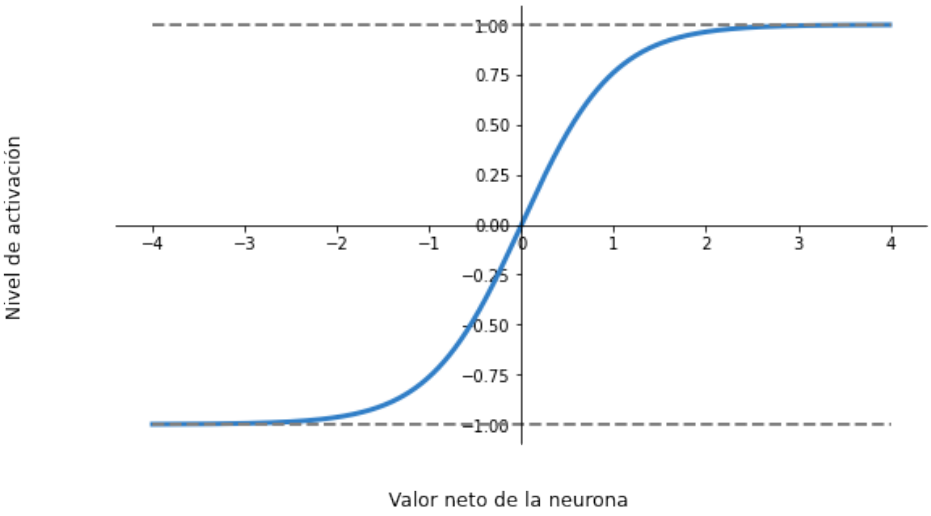

Figura 2.3: Función de activación *tanh* [48].

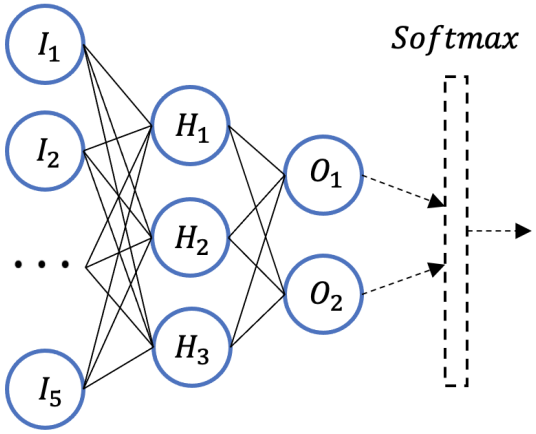

Figura 2.4: Representación de una red neuronal con una capa de entrada, una oculta y una de salida [76].

2.5.1. Redes convolucionales

Las redes convolucionales, también conocidas como redes neuronales convolucionales o CNNs, son un tipo de red neuronal especializada para procesar datos que tiene una topología conocida similar a un arreglo. Algunos ejemplos incluyen datos de series de tiempo, los cuales se pueden modelar como un arreglo 1D tomando muestras en un intervalo regular de tiempo. Otro ejemplo puede ser una imagen la cuál puede ser vista como un arreglo 2D de píxeles. Y también un texto el cuál se puede modelar como una secuencias de palabras.

El término red neuronal convolucional indica que la red emplea la operación matemática llamada convolución. Las redes convolucionales simplemente son redes neuronales que usan convolución en lugar de multiplicación general de matrices en al menos una de sus capas [36].

La arquitectura de una CNN está compuesta por capas convolucionales que aplican un conjunto de filtros a la entrada. Estos filtros buscan patrones específicos en la entrada, como bordes, esquinas o formas geométricas más complejas. Los resultados de estas operaciones son

luego reducidos en capas de pooling, que disminuyen la dimensión de la entrada y extraen características invariantes a pequeñas traslaciones. Finalmente, la salida de estas capas es procesada por capas totalmente conectadas, que aprenden a clasificar los datos de entrada en las diferentes categorías de salida [36].

Las redes neuronales convolucionales han tenido un gran éxito en aplicaciones de clasificación de imágenes [42], detección y reconocimiento de objetos [34], segmentación semántica de imágenes [71], reconocimiento de expresiones faciales [37], procesamiento de texto, análisis de sentimiento [57], reconocimiento de voz y procesamiento del habla [1].

2.5.2. Funciones de pérdida

Una función de pérdida es una medida utilizada para evaluar qué tan bien un modelo de aprendizaje automático se ajusta a los datos de entrenamiento. Una función de pérdida cuantifica la discrepancia entre las predicciones del modelo y las etiquetas verdaderas de los datos de entrenamiento [36].

La elección de una función de pérdida adecuada depende del tipo de problema que se está resolviendo.

Algunas funciones de pérdida comunes incluyen [8, 36]:

- Error cuadrático medio (MSE): es una función de pérdida utilizada comúnmente en problemas de regresión. Calcula el promedio de los cuadrados de las diferencias entre las predicciones del modelo y las etiquetas verdaderas.
- Entropía cruzada (*cross-entropy*): es una función de pérdida comúnmente utilizada en problemas de clasificación. Mide la discrepancia entre las probabilidades predichas por el modelo y las probabilidades verdaderas de las etiquetas.
- Pérdida de divergencia KL (*KL divergence loss*): es una función de pérdida utilizada comúnmente en problemas de aprendizaje no supervisado. Mide la distancia entre dos distribuciones de probabilidad, como la distribución de los datos de entrada y la distribución de los datos generados por el modelo.
- Entropía cruzada binaria (*Binary Cross-Entropy*): Esta función mide la diferencia entre dos distribuciones de probabilidad:
  - La distribución de probabilidad verdadera de los datos de entrenamiento.
  - La distribución de probabilidad estimada por el modelo.

  En términos más simples, la Entropía Cruzada Binaria se utiliza para medir qué tan bien un modelo de clasificación binaria puede predecir la etiqueta correcta para cada ejemplo en el conjunto de datos de entrenamiento. Cuanto menor sea el valor de la Entropía Cruzada Binaria, mejor será el modelo en la tarea de clasificación.

  La fórmula matemática de la Entropía Cruzada Binaria se expresa como:

$$BCE = -\sum_{i=1}^{N}[y^{(i)}log(\hat{y}^{(i)}) + (1 - y^{(i)})log(1 - \hat{y}^{(i)})] \tag{2.4}$$

  Donde, BCE representa la Entropía Cruzada Binaria. y es el valor verdadero de la etiqueta binaria (0 o 1). $\hat{y}$ es la probabilidad estimada de la etiqueta binaria (un número

entre 0 y 1) dada por el modelo. N es el número total de ejemplos de entrenamiento en el conjunto de datos [14].

La Entropía Cruzada Binaria se utiliza comúnmente en algoritmos de aprendizaje profundo, como las redes neuronales, durante el proceso de entrenamiento para ajustar los pesos y los sesgos de las neuronas para minimizar la función de costo.

La elección de una función de pérdida adecuada es crucial para el éxito del modelo de aprendizaje automático.

2.5.3. Funciones de activación

Una función de activación es una función matemática utilizada en las redes neuronales para determinar la salida de una neurona en función de su entrada. Es decir, es la función que se aplica a la suma ponderada de las entradas de una neurona para determinar su salida. Las funciones de activación son esenciales en la creación de redes neuronales profundas, ya que permiten la no linealidad en la salida de las neuronas [36].

Algunas funciones de activación más comunes son [8, 36]:

- Sigmoide: es una función matemática que toma un valor real como entrada y produce una salida en el rango de 0 a 1. Es una función suave y no lineal que se utiliza comúnmente en la capa de salida de una red neuronal para clasificación binaria.
- ReLU (*Rectified Linear Unit*): es una función de activación no lineal que produce una salida de 0 o un valor positivo. Esta función se utiliza comúnmente en las capas ocultas de una red neuronal debido a su capacidad para evitar el problema del desvanecimiento del gradiente.
- tanh (Tangente hiperbólica): es una función matemática que toma un valor real como entrada y produce una salida en el rango de -1 a 1, Figura 2.3. Es similar a la función sigmoide, pero su salida es simétrica en torno a cero.

2.6. Medidas de evaluación

Las métricas de evaluación son medidas que se utilizan para evaluar el rendimiento de los algoritmos de inteligencia artificial. Con estas métricas, es posible saber en qué medida un modelo dado está realizando de manera correcta una tarea en específico. A lo largo del tiempo se han propuesto e introducido métricas para evaluar en concreto los modelos propuestos para resolver la tarea de verificación de autoría. En seguida se listan las propuestas en el PAN 2022 (desafío de verificación de autoría) [28]:

- Exactitud (Accuracy): mide la proporción de todas las predicciones (tanto las predicciones correctas como las incorrectas) que el modelo clasifica correctamente. La exactitud es útil cuando las clases están equilibradas en el conjunto de datos de entrenamiento.
- Precisión: es la proporción de predicciones correctas sobre el total de predicciones realizadas. La precisión es una medida importante en la evaluación de modelos de clasificación, especialmente cuando las clases están desequilibradas [50].

- Sensibilidad (Recall): es la proporción de verdaderos positivos (instancias correctamente clasificadas) sobre el total de instancias que pertenecen a esa clase. Es útil cuando se desea minimizar los falsos negativos, es decir, cuando se desea identificar todas las instancias positivas [90].

- Medida F1 (F1 Score): es la media armónica de la precisión y Recall. Esta métrica es utilizada cuando se desea encontrar un equilibrio entre la precisión y la sensibilidad [9].

- Área bajo la curva ROC (AUC): el AUC mide la capacidad de un modelo para discriminar entre las clases. Esta métrica es utilizada cuando se desea comparar modelos que producen diferentes umbrales de probabilidad [32].

- c@1: es una extensión de la precisión, tiene un buen balance de discriminación, estabilidad y Recall. Esta medida es capaz de premiar a un sistema que mantiene el mismo número de respuestas correctas y al mismo tiempo decrece el número de incorrectas, al dejar algunas preguntas sin responder. Esta medida es adecuada para tareas como Comprensión de lectura, donde se proveen de múltiples respuestas por cada pregunta, pero solo una es la correcta [87].

- F_0.5u: Un problema con c@1 es que esa medida esta diseñada para clasificación binaria con igual peso para ambas clases. En el caso de la verificación de autoría, si se está interesado en decidir si dos textos fueron escritos por el mismo autor, pero no se necesita una decisión precisa en el otro caso, c@1 no sirve del todo. Por esta razón, se propone la medida F_0.5 donde se tratan la falta de respuesta como un falso negativo:

$$F_{0.5}u = \frac{(1+0.5^2)\cdot n_{tp}}{(1+0.5^2)\cdot n_{tp} + 0.5^2\cdot(n_{fn}+n_u) + n_{fp}} \tag{2.5}$$

  Donde n_{tp} denota el número de verdaderos positivos, n_{fn} el número de falsos negativos, y n_{fp} el número de falsos positivos. n_u es el número de preguntas no respondidas [7].

- Brier: es una métrica de evaluación para evaluar la habilidad de un modelo de clasificación para predecir probabilidades de eventos. Es una medida de error cuadrático medio (MSE) entre las probabilidades predichas y las probabilidades reales.

  La fórmula para calcular la medida Brier es la siguiente:

$$Brier = \frac{1}{N}\sum_{t=1}^{N}(p_i - o_i)^2 \tag{2.6}$$

  Donde N es el número total de observaciones o instancias. p_i es la probabilidad predicha por el modelo para la observación i. o_i es el resultado observado real para la observación i (1 para el evento que ocurrió, 0 para el evento que no ocurrió).

  La medida Brier es una métrica que se utiliza para evaluar modelos de clasificación en los que la probabilidad de ocurrencia de un evento es importante, como en problemas de calibración de modelos de riesgo o en la predicción de eventos extraños [12, 82].

Estas son algunas de las métricas de evaluación más comunes utilizadas en PLN. Sin embargo, la elección de la métrica adecuada dependerá del problema y de los objetivos de la tarea de clasificación.

2.7. Resumen

En este capítulo se expusieron la terminología y conceptos relacionados con el procesamiento de lenguaje natural, análisis de autoría, verificación de autoría, métodos de representación de textos, redes neuronales y medidas de evaluación de modelos de clasificación. Cabe destacar que cada uno de los temas expuestos en este capítulos de *Antecedentes* contiene un amplio campo de estudio, es por eso que, deben ser tratados como una introducción a estos temas. Para que sea posible entender el contenido del documento. Los temas en los que se requiera profundizar serán abordados en capítulos posteriores.

Capítulo 3

Verificación de Autoría

En este capítulo se presenta un panorama global del estado del arte en el área de análisis de autoría, en específico en la tarea de verificación de autoría. Durante la última década, se han propuesto una extensa lista de métodos para resolver la tarea de verificación de autoría [11, 27, 89, 98]. La efectividad de los enfoques para solucionar la tarea de verificación de autoría dependen de varios factores. Un de ellos es la longitud del texto usualmente la efectividad de los modelos se deteriora cuando se tienen textos cortos o muy cortos [98]. Otro caso realmente desafiante considera casos donde los textos de conocida y desconocida autoría pertenecen a diferente dominio, por ejemplo, un texto que hable de política contra otro que hable de deporte [28]. Así como estas variantes, existen otras que serán explicadas a lo largo del capítulo y principalmente en la que estará enfocada este documento, la variante de discurso cruzado.

Actualmente existen métodos para solucionar la tarea de verificación de autoría, los mas usados comprenden el uso de modelos que extraen características lingüísticas, y posteriormente un método de clasificación como las máquinas de vectores de soporte o redes neuronales. También se han aplicado modelos no supervisados, como la similitud coseno o compresión de textos [28].

3.1. Tarea de verificación de autoría

A medida que avanzan las investigaciones sobre esta tarea se han ido solucionando los escenarios más sencillos y controlados, por consecuencia, han comenzado a surgir escenarios más complicados que han llamando la atención de más investigadores por sus distintas aplicaciones; por ejemplo, en una variedad de investigaciones delitos informáticos que van desde homicidio hasta robo de identidad y muchos tipos de delitos financieros [21] o en el contexto de identificar al autor del código fuente [33].

La atribución de autoría trata de revelar información sobre la persona que escribió el texto [62, 97]. Dentro de este campo de estudio existen varias tareas para emular condiciones que pueden suceder en el mundo real, principalmente atribución de autoría de conjunto cerrado (donde hay un conjunto finito de autores) y atribución de autoría de conjunto abierto (donde hay un conjunto de candidatos pero no necesariamente incluye al verdadero autor o autores) [63]. El primer caso entra cuando una lista corta de personas pueden ser los autores de los textos en disputa mientras que el segundo caso entra cuando la lista de candidatos no es facilitada. La verificación de autoría es un caso especial de la atribución de autoría de conjunto abierto donde solamente hay un autor candidato [98]. En la verificación de autoría, textos de autoría conocida de un cierto autor son presentados a un sistema, y la tarea del sistema es

verificar si otro texto dado ha sido escrito por el mismo autor [38, 89]. En este caso, solo un texto de autoría conocida es dado al sistema [65]. Entonces por una pareja de textos (uno de autoría conocida y el otro de autoría desconocida), se necesita determinar si fueron escritos por el mismo autor.

Si bien la tarea de verificación de autoría ha sido bastante estudiada y ya existen modelos que resuelven este problema con muy buenos resultados. Hay variantes que pueden dificultar esta tarea, un simple ejemplo, es la longitud de los textos. Los modelos para resolver la tarea muestran deficiencia cuando se evalúan con textos cortos. Así como esta variante, se han propuesto diferentes casos que se pueden encontrar en el mundo real cuando se desea verificar la autoría de un texto. A continuación se definen algunas de las variantes de esta tarea.

En general, todas estas tareas emplean un análisis estilométrico de textos. Con esto, se crean huellas estilísticas que incluyen características léxicas como n-gramas de caracteres [96], frecuencia de palabras [45] o promedio de la longitud de palabra/oración [118], características sintácticas como categorías gramaticales (etiquetas POS) [108] o características estructurales tal como usos de sangría [118].

3.1.1. Verificación de autoría de dominio cruzado

En este caso, tanto los textos utilizados para entrenar un modelo como para evaluarlo pertenecen a diferentes dominios que pueden referirse a un tema, genero o lenguaje [56]. El escenario mas frecuente es el de tema cruzado. En el que, por ejemplo, los textos utilizados para entrenar un modelo podrían hablar de política mientras que los textos para evaluar el modelo podrían ser de deporte o algún otro tema. Otros escenarios importantes son el de género cruzado, en el que los conjuntos de texto pertenecen a distinto género literario (ensayos, reseñas, etc), y el de idioma cruzado en el que los conjuntos de texto se encuentran en diferente idioma (Ingles, Alemán, Francés, etc). En el PAN 2015 (reto de verificación de autoría) [56], tanto tema y género cruzado fueron considerados para la tarea de verificación de autoría y los resultados fueron relativamente bajos en exactitud, especialmente en genero cruzado en alemán [81].

3.1.2. Verificación de autoría de discurso cruzado

En la verificación de autoría de discurso cruzado se tienen textos de conocida y desconocida autoría que pertenecen a distinto tipo de discurso. En particular, estos discursos tienen significativas diferencias en cuanto al propósito comunicativo, destinado a cierta audiencia, o nivel de formalidad. Por ejemplo, el tipo de discurso de un ensayo argumentativo y un mensaje de texto enviado a un miembro de la familia tienen importantes diferencias estilísticas impuestas por las normas del tipo de discurso [28].

Es por eso que en este caso resulta complicado distinguir las características autorales que permanecen intactas en todos los tipos de discurso. Además, los tipos de discurso se correlacionan fuertemente con la longitud del texto (por ejemplo, los ensayos son mucho mas largos que los mensajes de texto). También la verificación de autoría de discurso cruzado puede ser usada para estudiar el efecto de la longitud del texto en la eficiencia de los enfoques para resolver la tarea [28].

3.2. Métodos de verificación de autoría

En general, el enfoque más popular en el análisis de autoría es la extracción de características del texto y usarlas para entrenar un algoritmo de clasificación. El método de extracción

de características podría estar determinado por los requisitos computacionales en el nivel semántico, es decir, dependencias semánticas, sinónimos; nivel sintáctico, es decir, fragmentos, etiquetas, POS, estructura de oraciones y frases; nivel de caracter, es decir, tipo de caracter, n-gramas de caracter, número de caracteres especiales; y nivel léxico, es decir, palabras mal escritas, longitud de oración, longitud de palabra, bolsa de palabras, riqueza de vocabulario [97].

Algunos algoritmos de clasificación supervisados usados en verificación de autoría son máquinas de vectores de soporte, árboles de decisión, análisis discriminante, redes neuronales, y algoritmos genéticos [100]. Otra opción es calcular la similitud entre los textos y usarla para predecir si los dos textos son escritos por el mismo autor o no [65].

Una solución que se diseñó específicamente para la tarea de verificación de autoría es el método de des-enmascaramiento, propuesto en [64]. Este enfoque trata de medir que tan profunda es la diferencia entre dos textos comparándolos muchas veces, y en cada iteración usar menos características relevantes en uno de ellos. Este método logró obtener 95.7 % de exactitud cuando se evaluó con textos largos (textos de alrededor de 500 mil palabras). Cuando se evaluó el mismo método pero utilizando textos cortos o textos de distinto género y tema, el rendimiento decrecía considerablemente. Por ejemplo, en [54] evalúan el rendimiento del método de des-enmascaramiento usando un conjunto de textos teatrales y en prosa de unas 10,000 palabras obteniendo una exactitud de solo 77 %. En [7] se propuso una alternativa del método de des-enmascaramiento que obtiene una exactitud entre el 75 % y 80 % utilizando textos cortos de alrededor 4000 palabras. El rendimiento de el método de des-enmascaramiento original depende de la disponibilidad de suficientes párrafos por texto, cada uno de suficiente extensión. Si los párrafos son demasiado cortos, los datos de entrenamiento se vuelve demasiado escaso y no se pueden generar curvas descriptivas.

Otro método relevante para resolver la tarea de verificación de autoría es el método del impostor [63, 65]. Este método propone usar un conjunto de documentos impostores recolectados de la web de manera que estos documentos tengan un tema similar con el documento conocido y desconocido. También se define un conjunto de características como las palabras de función, n-gramas de palabras y n-gramas de caracteres.

Antes del año 2015, la tarea de verificación de autoría del PAN [102] fue evaluada utilizando principalmente conjuntos de texto del mismo género y tema tanto en entrenamiento como en prueba. Se observó que en escenarios de tema cruzado el rendimiento de los métodos tradicionales decrece [102].

En [103] se mostró que los n-gramas de caracteres son mas confiables que otras características léxicas para la tarea de análisis de autoría cuando el género y el tema de los documentos varían. En [92] se especifica que algunos tipos de n-gramas funcionan mejor que otros como características para la clasificación. En particular, algunos trigramas que incluyen signos de puntuación funcionan mejor en un escenario cruzado.

PAN es una serie de eventos científicos y tareas compartidas sobre análisis forense de texto digital y estilometría (https://pan.webis.de/, acceso en Marzo 2023). Desde 2011, se han enfocado en tareas de análisis de autoría y los métodos presentados en su taller son el estado del arte de este campo de estudio.

3.2.1. Métodos basados en la distancia

Los métodos basados en la distancia están diseñados mediante el uso de una función de similitud para medir la cercanía entre los objetos de texto. La función de similitud más conocida que se usa comúnmente en el dominio del texto es la función de similitud coseno.

3.2.1.1. Similitud coseno

Es una medida utilizada para evaluar la similitud entre dos textos muy utilizada en PLN. Este método se basa en la idea de que los textos similares tienen un contenido semántico similar y comparten palabras en común.

Para calcular la similitud coseno entre dos textos, se utiliza la representación vectorial de los textos en un espacio de n-dimensiones. Entonces, sea $U = (f(u_1)...f(u_k))$ y $V = (f(v_1)...f(v_k))$. Donde los valores $u_1...u_k$ y $v_1...v_k$ representan las frecuencias de los términos (normalizados), y la función $f(\cdot)$ representa la función de amortiguamiento (*damping function*). Las funciones de amortiguamiento típicas para $f(\cdot)$ podrían representar la raíz cuadrada o el logaritmo [18]. Entonces la similitud coseno entre los dos documentos se define con la Ecuación 3.1 y en la Figura 3.1 se ilustra la distancia entre los documentos en un espacio de dos dimensiones.

$$simcos(\vec{u}, \vec{v}) = \frac{\vec{u} \cdot \vec{v}}{\|\vec{u}\|\|\vec{v}\|} = \frac{\sum_{i=1}^{k} f(u_i) \cdot f(v_i)}{\sqrt{\sum_{i=1}^{k} f(u_i)^2} \cdot \sqrt{\sum_{i=1}^{k} f(v_i)^2}} \tag{3.1}$$

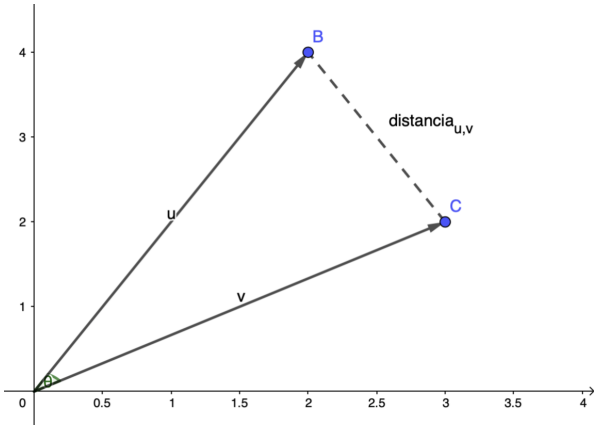

Figura 3.1: Distancia entre dos documentos representados por los vectores $\vec{u}$ y $\vec{v}$ [18].

Este método ofrece una solución ingenua pero rápida para la tarea de verificación de autoría. Todos los documentos pueden ser representados usando un modelo de bolsa de gramas de caracteres, que está ponderado por TF-IDF. La similitud coseno entre cada par de documentos en el conjunto de datos de entrenamiento es calculado. Finalmente, las similitudes resultantes son optimizadas, y proyectadas a través de una simple operación de reescalado, para que puedan funcionar como pseudo-probabilidades, indicando la probabilidad de que un par de documentos sea un par del mismo autor. A través de una búsqueda en cuadrícula, se determina el umbral de verificación óptimo, teniendo en cuenta que algunos problemas difíciles pueden quedar sin respuesta [28].

3.2.1.2. Distancia Delta

Para determinar o verificar el autor de un texto, Burrows [16] sugiere que los textos similares tendrán una distribución de palabras similar, pero la distribución de las palabras puede variar debido a factores como el estilo y la temática. Este método utiliza la distancia del coseno para

medir la similitud entre los perfiles de frecuencia de las palabras en los textos. El perfil de frecuencia de una palabra es una lista de las frecuencias de ocurrencia de esa palabra en cada uno de los textos.

La distancia Delta se utiliza para medir la diferencia entre los perfiles de frecuencia de las palabras en dos textos. Esta distancia es una medida de la diferencia entre dos perfiles de frecuencia que tiene en cuenta tanto la cantidad como la calidad de las palabras que aparecen en los textos [31].

Al comparar la distancia Delta entre un texto desconocido y un conjunto de textos de referencia conocidos, se puede determinar la similitud entre los textos y evaluar si el texto desconocido es una copia o una obra original.

3.2.1.3. Divergencia de Kullback-Leibler (KL)

El método de divergencia de Kullback-Leibler (KL) es una técnica que se utiliza para medir la distancia entre dos distribuciones de probabilidad.

Representar el estilo específico de un autor podría significar contar las frecuencias relativas de los tipos de palabras más frecuentes. Pero si los tipos de palabras mas frecuentes de un lenguaje dado corresponden a las palabras funcionales, no es necesario definirlas según un corpus. Se podría simplemente definirlas antes de alguna investigación. Siguiendo este punto de vista, Zhao y Zobel [117] sugieren considerar un número limitado de tipos de palabras predefinidos para discriminar entre varios autores. Propusieron una lista fija de palabras en ingles que contiene 363 términos que cubren principalmente las funciones (por ejemplo., the, in, but, not, am, of, can), y también ciertas formas que aparecen con frecuencia (por ejemplo., became, nothing). Otras en este lista de palabras no son muy frecuentes (por ejemplo., howbeit, whereafter, whereupon), mientras que algunas revelan el comportamiento esperado del tokenizador (es decir., doesn, weren) o parecen corresponder a ciertas decisiones arbitrarias (es decir., indicate, missing, specifying, seemed). Como una alternativa, se puede aplicar la propuesta por Antonia et al. [2] una lista que contiene 192 entradas que corresponden a palabras funcionales que aparecen en el inglés moderno e inglés victoriano. Después de definir este conjunto de características fijas, se debe estimar la probabilidad de ocurrencia de cada término o característica para un perfil de autor o un textos en disputa. A partir de estas estimaciones se puede evaluar el grado de discrepancia entre las dos distribuciones probabilísticas. Para lograr esto, Zhao and Zobel [117] propusieron usar la fórmula de divergencia de Kullback-Leibler (KLD), también llamada entropía relativa. EL valor KLD se expresa en la Ecuación 3.2 e indica en qué medida la distribución de características derivada del texto de consulta Q diverge de la j-ésima distribución del perfil de autor A_j

$$KLD(Q||A_j) = \sum_{i=1}^{m} p(t_i, Q) \cdot log_2(\frac{p(t_i, Q)}{p(t_i, A_j)}) \tag{3.2}$$

Donde $p(t_i, Q)$ y $p(t_i, A_j)$ indican la probabilidad de ocurrencia del término t_i en el texto en cuestión Q o en el j-ésimo perfil del autor, respectivamente. En este cálculo, se supone que $0 \cdot log_2(0/p) = 0$, y $p \cdot log_2(p/0) = \infty$

Con esta definición, y cuando las dos distribuciones son idénticas, el valor resultante es cero, mientras en todos los otros casos el valor resultante es positivo. Por ejemplo, en la Tabla 3.1, se reportan tres perfiles de autores, así como el sustituto estilístico del texto Q basado solo en tres términos. Un análisis rápido a los datos representados en esta tabla muestra que el autor

A_3 es el más cercano al texto en disputa, mientras que el autor A_1 es el más lejano. El estilo del autor A_2 representa una distribución uniforme sobre los tres términos.

	Término t_1	Término t_2	Término t_3
A_1	0.1	0.2	0.7
A_2	0.333	0.333	0.333
A_3	0.45	0.35	0.2
Q	0.5	0.3	0.2

Tabla 3.1: Representación de tres perfiles de autores junto con un texto en disputa Q [48].

En la Tabla 3.2, cada celda indica la contribución de cada término según el perfil de cada autor al considerar el texto en disputa Q. El puntaje general se indica en la última columna de la Tabla 3.2 (bajo la etiqueta "KLD"). Cuando las probabilidades estimadas para un término son las mismas en el texto de la consulta y en el perfil, el impacto es nulo. Cuando esta estimación es mayor en el texto en disputa que en el perfil del autor, el valor calculado es positivo (y negativo en caso contrario).

	Término t_1	Término t_2	Término t_3	KLD
$KLD(Q\|\|A_1) =$	1.161	0.175	-0.361	0.975
$KLD(Q\|\|A_2) =$	0.293	-0.046	-0.147	0.100
$KLD(Q\|\|A_3) =$	0.076	-0.067	0.000	0.009

Tabla 3.2: Contribución de cada componente y valor KLD final [48].

Con este enfoque, la principal preocupación es estimar con precisión las probabilidades de ocurrencia.

En resumen, el método de divergencia de Kullback-Leibler es una técnica utilizada para medir la distancia entre dos distribuciones de probabilidad. La KLD se utiliza para tareas como la verificación de autoría, recuperación de información, clasificación de textos y la agrupación de documentos, y permite medir la similitud entre textos y evaluar la relevancia de los resultados de búsqueda [48].

3.2.2. Métodos basados en la compresión de textos

Los métodos tradicionales utilizados en verificación de autoría comparten algunos problemas: la necesidad de extraer características antes del procesamiento; la necesidad de definir los límites entre palabras y la necesidad de lidiar con las variantes morfológicas de las palabras. Los métodos de clasificación de texto basados en compresión son fáciles de aplicar y prácticamente no requieren preprocesamiento de los datos y son igualmente efectivos aplicados a la atribución y verificación de autoría [91]. La mayoría de estos métodos se basan en caracteres y, por lo tanto, tienen el potencial de capturar automáticamente las características que no son palabras de un documento, como la puntuación, las raíces de las palabras y las características que abarcan más de una palabra [53].

3.2.2.1. Modelo de compresión

En [107] proponen usar un modelo de lenguaje desarrollado para la compresión de texto como base de un esquema de categorización de texto y potencialmente para otras aplicaciones

como segmentación de palabras, minería de textos, identificación de lenguaje y verificación de autoría.

Este método gira en torno a la noción de entropía como una medida de contenido de información de un mensaje [107], y lo que es más importante, que la compresión de texto se puede usar directamente para estimar un límite superior para la entropía [13].

Formalmente, suponiendo que se tiene un lenguaje definido por un alfabeto de k símbolos s_i, y una distribución de probabilidad sobre estos símbolos $p(s)$. El teorema fundamental de codificación establece que el límite inferior en el número promedio de bits por símbolo necesarios para codificar el mensaje del idioma (es decir, secuencia de símbolos) está dado por la entropía de la distribución de probabilidad (y regularmente referida como entropía de lenguaje 3.3).

$$H(language) = H(p) = -\sum_{i=1}^{k} p(s_i) log_2 p(s_i) \tag{3.3}$$

El contenido de información de un símbolo en particular viene dado por $-log_2 p(s_i)$, y es el número de bits necesarios para codificar ese símbolo. Por lo tanto, $H(p)$ puede verse como una expectativa sobre el número de bits por símbolo. Por lo tanto, podemos obtener longitudes de código mínimas o, alternativamente tasas de compresión máximas, codificando símbolos de acuerdo con la distribución de probabilidad real $p(s)$.

En general, no se tiene acceso a la distribución de probabilidad real subyacente a un idioma, sino a algún modelo de ese idioma. Suponiendo que se tiene un lenguaje, L, en el que la secuencia de símbolos $x_1, x_2, ..., x_n$ (denominada x_{1n} en adelante) se generan de acuerdo con la distribución de probabilidad $p(x)$, para la cual se construye un modelo $P_M(x)$. La entropía cruzada del lenguaje con respecto al modelo proporciona una medida de la extensión en que el modelo se aparta de la distribución real y viene dada por:

$$H(L, p_M) = -\lim_{x \to \infty} \frac{1}{n} \sum_{x_{1n}} p(x_{1n}) log_2 p_M(x_{1n}) \tag{3.4}$$

La entropía cruzada es el número promedio de bits por símbolo necesarios para codificar el lenguaje L utilizando el modelo p_M y, por lo tanto, $H(L, p_M)$ es un límite superior de la entropía del lenguaje y $H(L) \leq H(L, p_M)$. $H(L, p_M)$ puede aproximarse observando que la Ecuación 3.4 es simplemente el valor esperado de $-log_2 p_M(x_{1n})$ sobre una muestra infinita de ejemplos de lenguaje, y se aproxima usando una muestra n muy grande de la siguiente manera:

$$H(L, p_M) \approx -\frac{1}{n} log_2 p_M(x_{1n}) \tag{3.5}$$

Para un documento dado, D, se puede calcular la entropía cruzada del documento, el número promedio de bits por símbolo necesarios para codificar el documento usando el modelo como:

$$H(L, p_M, D) = -\frac{1}{n} log_2 p_M(D) \tag{3.6}$$

Suponiendo que p_M es un buen modelo para L, entonces cuanto menor sea la entropía cruzada del documento, más probable es que el documento D sea generado por el lenguaje.

3.2.2.2. Modelo de lenguaje basados en PPM

El esquema de compresión de texto por predicción por coincidencia parcial (PPM por las siglas en inglés) ha establecido constantemente el estándar en compresión de texto sin pérdida desde que se publicó originalmente en 1984 por Cleary and Witten [23].

Los modelos de compresión codifican un documento de acuerdo con un modelo dado, y la entropía cruzada resultante viene dada por la Ecuación 3.7. En el caso de la familia de modelos PPM, los símbolos individuales se codifican dentro del contexto proporcionado por los símbolos anteriores que aparecen en un documento. Tomando la Ecuación 3.7. como punto de partida, se puede lograr una compresión óptima para el modelo p_M, al observar que:

$$H(L, p_M, D) = \frac{1}{n} \sum_{i=1}^{n} -log_2 p_M(x_i|context_i) \tag{3.7}$$

Donde $context_i = x_1, x_2, ..., x_{i-1}$. Así, cada símbolo se codifica de acuerdo con su contenido de información dentro del contexto proporcionado por todos los símbolos precedentes. En la práctica, PPM usa una aproximación de Markov y asume un contexto de orden fijo. El método PPM se basa en caracteres, aunque el mecanismo de combinación se puede aplicar igualmente a otras clases de símbolos, como palabras y partes del discurso. Los experimentos de compresión con una amplia gama de texto en inglés [106] muestran que los modelos basados en palabras superan consistentemente a los métodos basados en caracteres, aunque la diferencia en el rendimiento es solo de un pequeño porcentaje. En [75] presentan resultados de una serie de experimentos diseñados para evaluar la efectividad y el comportamiento de diferentes métodos de clasificación de texto basados en compresión en texto en inglés.

Utilizando los beneficios que ofrece el modelos de compresión de texto, es posible modelar un método para verificación de autoría basado en este principio. En [40] describen un modelo de identificación de autor que aunque propiamente no es una tarea de verificación de autoría, se puede utilizar el mismo modelo para verificar si ambos textos fueron o no escritos por el mismo autor. Dicho modelo se usa junto con un etiquetador POS. Utilizan etiquetas POS específicas del idioma estándar en los textos en cuestión para generar un flujo de símbolos que representan cada palabra, posteriormente se ejecuta el algoritmo PPM en el flujo resultante y se utiliza el método de Bobicev [10] para calcular y comparar las entropías cruzadas para determinar la autoría del texto.

3.2.3. Métodos basados en aprendizaje automático

Cada estudio estilométrico o de lingüística cuantitativa puede seguir la siguiente ruta subdividida en seis pasos principales. Primero, se debe formular una pregunta o hipótesis de investigación precisa de acuerdo con una teoría o para verificar una hipótesis. En segundo lugar, se debe recopilar una muestra de textos para crear un corpus de evaluación. En tercer lugar, se deben aplicar algunos procedimientos de preprocesamiento para controlar la calidad de los datos, eliminar elementos extratextuales y, por lo general, normalizar la ortografía (un significado = una ortografía). Cuarto, se debe elegir una estrategia de representación de texto para extraer las características estilísticas pertinentes que reflejen una categoría o un estilo de escritura personal. En quinto lugar, se podrían emplear varios procedimientos de selección de

características para eliminar los atributos ruidosos, reducir el costo computacional y disminuir el riesgo de sobreajuste. En la sexta etapa, se elige un algoritmo de aprendizaje automático y se aplica al conjunto de datos.

Esta sección presenta enfoques base utilizados en PLN para resolver tareas de verificación de autoría, atribución de autoría, clasificación de textos, entre otras.

3.2.3.1. k vecinos más cercanos (k-NN)

El modelo de vecino más cercano (NN por sus siglas en inglés) representa cada instancia como un vector (o un punto) en un espacio de m dimensiones donde cada dimensión corresponde a una característica (o un término). Como ejemplo, la Tabla 3.3 muestra la representación de ocho artículos en cuatro dimensiones, usando las preposiciones *by*, *upon*, *on* y el verbo modal *would*. Los tres posibles autores aparecen con dos textos y las dos últimas columnas representan la representación de dos artículos en disputa, $Q54$ y $Q55$. En esta tabla, cada celda indica el porcentaje de tokens que se producen en el artículo correspondiente. En el artículo $H8$, por ejemplo, se puede ver un 0,55 % de las fichas correspondientes a la preposición *by* y un 1,36 % al verbo modal *would*. Como estos porcentajes indican las frecuencias relativas, se pueden interpretar como una estimación de su probabilidad de ocurrencia [48].

Termino	$H8$	$H59$	$M38$	$M40$	$J2$	$J64$	$Q54$	$Q55$
by	0.55 %	0.90 %	1.11 %	1.82 %	0.60 %	1.30 %	1.30 %	0.69 %
upon	0.15 %	0.16 %	0.12 %	0.00 %	0.06 %	0.00 %	0.10 %	0.00 %
on	0.45 %	0.32 %	0.45 %	0.43 %	0.48 %	0.61 %	0.95 %	0.44 %
would	1.36 %	0.84 %	0.45 %	0.20 %	0.30 %	0.30 %	0.30 %	0.49 %

Tabla 3.3: Porcentaje de cuatro tipos de palabras seleccionadas en artículos periodísticos escritos por tres autores y en dos artículos en disputa [48].

Un paso importante de preprocesamiento con el modelo NN es garantizar que cada atributo se mida en las mismas unidades o en unidades muy similares. Para asignar a todos los atributos un impacto similar en la distancia intertextual, es necesario normalizarlos. A continuación se muestra un método de normalización, otros métodos se pueden encontrar en [48]:

Teniendo n instancias en una muestra, el valor del atributo actual denotado a_i puede ser reemplazado por su valor normalizado denotado a_i de acuerdo con la Ecuación 3.8. En esta variante, para un indicador dado, cada valor se divide por la suma de todos los n valores.

$$a'_i = \frac{a_i}{\sum_{j=1}^{n} a_j} \tag{3.8}$$

Después de normalizar el conjunto de datos, se puede aplicar el modelo de aprendizaje. El enfoque NN no genera una representación aprendida específica, sino que utiliza directamente las instancias que pertenecen al conjunto de entrenamiento. Al limitar el número de atributos a dos, se pueden visualizar claramente los artículos y las distancias subyacentes entre ellos. En la Figura 3.2, el eje horizontal indica la frecuencia relativa de la palabra *by* y *would* es representada por el eje vertical.

En este diagrama de dispersión, los seis textos escritos por tres autores se insertan según los valores representados en la Tabla 3.3. Dentro de este pequeño conjunto de entrenamiento, se agregan los dos artículos en disputa (Q54 y Q55). Tomando como ejemplo Q55, se puede

calcular su distancia a todos los textos pertenecientes al conjunto de entrenamiento para determinar su vecino más cercano. Para lograr esto, la función euclidiana parece la única opción [48].

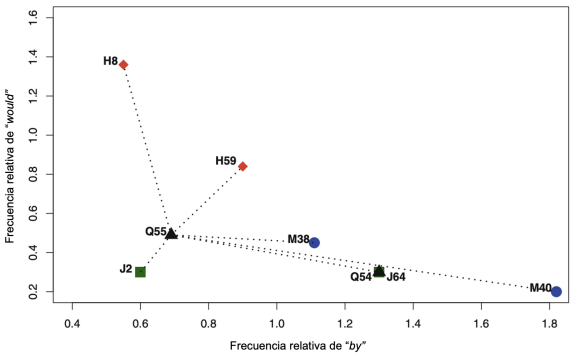

Figura 3.2: Modelo K-NN con dos atributos *by* y *would*. Imagen tomada de [48].

Después de elegir una función de distancia, el clasificador podría basarse en el método 1-NN. Para determinar este, se debe calcular la distancia con todas las instancias en la muestra de entrenamiento, y la más pequeña indica el vecino más cercano. Limitar la decisión a una sola instancia podría ser problemático, especialmente cuando el conjunto de datos podría tener ruido. Además, un clasificador efectivo debe poder generalizar la información proporcionada por el conjunto de entrenamiento. Tomar una decisión sobre una sola observación podría implicar un riesgo de sobreajuste de los resultados en los datos de entrenamiento. Por lo tanto, se recomienda considerar no un único vecino sino k de ellos [48].

Al aplicar un enfoque $k - NN$, la distancia de las k instancias más cercanas en la muestra de entrenamiento se tiene en cuenta para proporcionar la asignación final. La decisión final se toma por voto ponderado. Por supuesto, definir un valor de k eficiente no es evidente y se deben evaluar diferentes soluciones para verificar su efectividad. El costo del cálculo es una desventaja de este método. Para identificar el $k - NN$, se debe calcular la distancia a todos los sustitutos de texto que ocurren en el conjunto de entrenamiento (enfoque basado en instancias).

Halvani en [39] propone un método de verificación de autoría basado en k-vecino más cercano (k-NN) para la tarea de identificación de autor (AI) del desafío PAN 2013 [52]. El método sigue una técnica de clasificación de conjuntos basada en la combinación de categorías de características adecuadas. Para cada categoría de característica elegida, se aplica un clasificador k-NN para calcular una puntuación de desviación de estilo entre los documentos de formación del verdadero autor A y el documento de un autor, que afirma ser el autor A. Según la puntuación y un umbral dado, se genera una decisión a favor o en contra del presunto autor y se almacena en una lista. Posteriormente, la decisión final sobre la supuesta autoría se determina por mayoría de votos entre todas las decisiones dentro de esta lista. El método proporciona una serie de beneficios como, por ejemplo, la independencia de recursos lingüísticos como ontologías, tesauros o incluso modelos de lenguaje. Otro beneficio es la independencia lingüística entre los diferentes idiomas indoeuropeos, ya que el enfoque es aplicable a idiomas como el español, el inglés, el griego o el alemán. Otro beneficio es el bajo tiempo de ejecución del método, ya que no es necesario un procesamiento lingüístico profundo como el etiquetado POS, la fragmentación o el análisis. Además, el método se puede ampliar o modificar, por ejemplo, reemplazando la función de clasificación, el umbral o las características subyacentes,

incluidos sus parámetros (por ejemplo, tamaños de n-Gramos o la cantidad de frecuencias de características). Este modelo obtuvo una puntuación de precisión general del 80 % en el PAN 2013 [52].

3.2.3.2. Naive Bayes

El modelo bayesiano ingenuo multinomial es un clasificador de texto típico derivado del paradigma de aprendizaje automático. Con este enfoque, cada posible categoría (o autor) se denomina hipótesis y se denota como H_j para $j = 1, 2, ...r$. Para definir la categoría más probable de un texto de consulta Q, el modelo ingenuo de Bayes selecciona la que maximiza la Ecuación 3.9, en el que $t_{i,Q}$ representa el término i-ésimo en la secuencia de términos que aparecen en el texto de consulta Q, y n_Q indica la longitud de este texto de consulta. Esta ecuación combina dos estimaciones, la prioridad (denotada por $p(H_j)$) y la verosimilitud para determinar el máximo a posteriori (MAP por sus siglas en inglés) [48].

$$H_{MAP} = max_{H_j} p(H_j|Q) \propto \underbrace{p(H_j)}_{previa} \cdot \underbrace{\prod_{i=1}^{n_Q} p(t_{i,Q}|H_j)}_{probabilidad} \tag{3.9}$$

$$H_{MAP} = max_{H_j} p(H_j|Q) \propto p(H_j) \cdot \prod_{i=1}^{m} p(t_i|H_j)^{tf_{i,j}} \tag{3.10}$$

Como una vista alternativa, la verosimilitud puede ser calculada como se muestra en 3.10. donde cada $p(t_i|H_j) = p(t_{i,Q}|H_j)$ refleja la misma probabilidad de ocurrencia presente en la Ecuación 3.9 pero a la potencia $tf_{i,j}$. En la Ecuación 3.9, la multiplicación se realiza para cada término de la secuencia de palabras que aparece en el texto de consulta. Por ejemplo, si la palabra *while* aparece cinco veces en un texto, su probabilidad de ocurrencia ocurrirá cinco veces en la Ecuación 3.9, pero sólo una vez, a la quinta potencia, en la Ecuación 3.10. Por lo tanto, la multiplicación se realiza sobre todas las fichas en la Ecuación 3.9 y solo para todos los tipos de palabras distintos (indicados por m) en la Ecuación 3.10 [48].

Como ejemplo, en la Tabla 3.4, se muestra el contenido de cinco tweets hipotéticos que forman el conjunto de entrenamiento de un corpus. Se deben identificar dos categorías, textos escritos por un autor masculino (etiquetado como M) o femenino (F). Para estimar la probabilidad previa asociada con ambas categorías, se opta por una probabilidad previa no informativa ($p(M) = p(F) = 1/2$) o según el número de tweets en cada categoría ($p(M) = 2/5, p(F) = 3/5$).

Tweet ID	Texto	Categoría
1	football tv money tv	M
2	sport tv football sport friends	M
3	family	F
4	family friends family	F
5	friends tv sport	F
Q	friends friends tv	?

Tabla 3.4: Cinco tweets escritos por hombres (M) y mujeres (F) [48].

Para estimar la probabilidad de ocurrencia de cada palabra en cada categoría, se concatenan todos los tweets según cada clase. Para los escritores hombres, la longitud del texto resultante es 9 y para las mujeres 7. Para estimar la probabilidad de aparición de la palabra *tv* sabiendo que el escritor pertenece a la categoría M, se calcula la relación entre la frecuencia del término en esa clase ($tf_{i,j} = 3$) y la longitud del texto de esta categoría ($n_M = 9$), resultando una estimación de 3/9. La Tabla 3.5 muestra tanto la estimación directa para todas las palabras como para ambas categorías. Además, las dos últimas columnas muestran las mismas estimaciones con el suavizado de Laplace [48]. En este último caso, el numerador se incrementa en uno, y el denominador se suma por el número de palabras distintas en este corpus ($m = 6$).

	Direct		Laplace	
Word	$p(t_i\|M)$	$p(t_i\|F)$	$p(t_i\|M)$	$p(t_i\|F)$
family	0/9	3/7	$(0+1)/(9+6)$	$(3+1)/(7+6)$
friends	1/9	2/7	$(1+1)/(9+6)$	$(2+1)/(7+6)$
tv	3/9	1/7	$(3+1)/(9+6)$	$(1+1)/(7+6)$
football	2/9	0/7	$(2+1)/(9+6)$	$(0+1)/(7+6)$
money	1/9	0/7	$(1+1)/(9+6)$	$(0+1)/(7+6)$
sport	2/9	1/7	$(2+1)/(9+6)$	$(0+1)/(7+6)$

Tabla 3.5: Estimaciones de probabilidad de ocurrencia para la categoría hombres (M) y mujeres (F) [48].

Con base en las estimaciones que se muestran en la Tabla 3.5, se puede calcular la verosimilitud de observar el tweet de consulta Q que se muestra en la última fila de la Tabla 3.4. Este tweet en disputa contiene tres tokens (pero dos tipos de palabras). Utilizando una estimación directa y siguiendo la Ecuación 3.10, la verosimilitud para la categoría M se proporciona mediante la Ecuación 3.11 y para la otra clase por la Ecuación 3.12.

$$p(friend|M)^2 \cdot p(tv|M)^1 = (\frac{1}{9})^2 \cdot \frac{3}{9} = 0.0041 \tag{3.11}$$

$$p(friend|F)^2 \cdot p(tv|F)^1 = (\frac{2}{7})^2 \cdot \frac{1}{7} = 0.0117 \tag{3.12}$$

Utilizando una distribución uniforme para la probabilidad previa, la Ecuación 3.13 combina la probabilidad previa y la verosimilitud para la categoría masculina, y la Ecuación 3.14 para la femenina.

$$Male : \frac{1}{2} \cdot ((\frac{2}{15})^2 \cdot \frac{4}{15}) = 0.00237 \tag{3.13}$$

$$Female : \frac{1}{2} \cdot ((\frac{3}{13})^2 \cdot \frac{2}{13}) = 0.0040965 \tag{3.14}$$

Estos valores no son directamente estimaciones de probabilidad sino que son proporcionales a las probabilidades correspondientes. Para calcular las probabilidades exactas, la Ecua-

ción 3.15 indica que se debe tener en cuenta un factor de normalización (denominador de la Ecuación 3.15) [48].

$$p(H_j|Q) = \frac{p(H_j) \cdot \prod_{i=1}^{m} p(t_i|H_j)^{tf_{i,j}}}{\sum_{k=1}^{r} p(H_k) \cdot \prod_{i=1}^{m} p(t_i|H_k)^{tf_{i,k}}} \tag{3.15}$$

Este factor de normalización es la suma de todas las categorías posibles (para garantizar que la suma de todos los resultados posibles devuelva 1.0). En este caso, las probabilidades resultantes se proporcionan en las Ecuaciones 3.16 y 3.17, lo que indica que la probabilidad de que el tweet de consulta haya sido escrito por una mujer es mayor que la de un hombre.

$$p(Male|Q) = \frac{0.00237}{0.00237 + 0.0040965} = 0.3665 \tag{3.16}$$

$$p(Female|Q) = \frac{0.0040965}{0.00237 + 0.0040965} = 0.6335 \tag{3.17}$$

La eficacia del modelo Naive Bayes suele ser relativamente alta y podría utilizarse como referencia para verificar el rendimiento de modelos de aprendizaje más complejos.

3.2.3.3. Máquinas de vectores de soporte (SVM)

El modelo de máquina de vectores de soporte (SVM), es un enfoque popular en el aprendizaje automático y considerado como un enfoque eficaz para resolver los problemas de verificación de autoría, atribución de autoría y creación de perfiles de autor [53]. Para ejemplificar el funcionamiento del modelo en [48] utilizan los textos de un par de autores del corpus *Federalist Papers*, Hamilton con 45 artículos y Madison con 14. Los aspectos estilísticos de estos artículos están representados por las frecuencias relativas de solo dos preposiciones *to* y *upon* (computadas en %). Esta limitación permite visualizar todos estos artículos en dos dimensiones.

En la Figura 3.3, los artículos de Hamilton se muestran con un diamante rojo y cada círculo azul indica un artículo de Madison. Como esta figura se enfoca en una fracción del espacio destacado, pocos artículos no son visibles. Los textos de Hamilton presentan una alta frecuencia para ambas preposiciones y por eso aparecen arriba a la derecha. En contraste, los textos de Madison aparecen en la parte inferior izquierda, reflejando bajas frecuencias en el uso de *to* y *upon*.

Con el modelo SVM más simple, también llamado *clasificador de vectores de soporte*, el esquema de aprendizaje debe definir un borde lineal para separar todas las instancias de ambos autores (o categorías). Dado que en este ejemplo incluye dos atributos, este límite de clase corresponde a una línea. Cuando están presentes tres atributos, un plano determina el borde, y con más de tres características, se debe definir un hiperplano [48].

Para definir este límite de clase, SVM no necesita todas las instancias (o todas las representaciones de texto), sino solo un conjunto reducido denominado S que contiene todos los *vectores de soporte* (o puntos). En la Figura 3.3, el borde se determina con las representaciones de texto etiquetadas como H59, M38 y M40.

Tomando en cuenta la Figura 3.3, se pueden dibujar muchas líneas para dividir perfectamente todas las instancias según su autor. Así, la separación perfecta de las dos clases no es

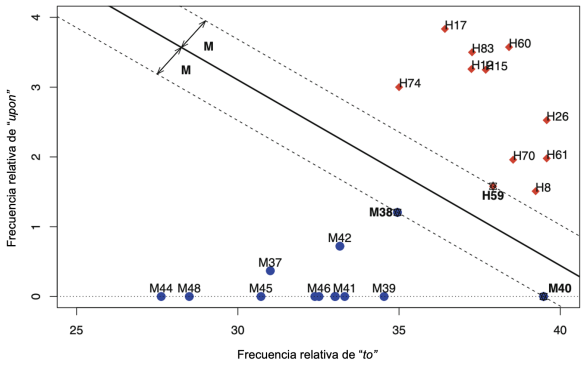

Figura 3.3: Modelo SVM con dos atributos *to* y *upon*. Imagen tomada de [48].

el criterio único y obligatorio. Es más importante que la frontera de clase esté lo más alejada posible de todos los puntos para generalizar bien la distinción entre los dos estilos.

Para formalizar este concepto, el margen se define como la distancia entre los puntos más cercanos y el límite de clase. Esta distancia se indica en la Figura 3.3 con la etiqueta "M". La misma distancia aparece en el lado izquierdo y en el lado derecho. Al determinar este límite de clase, SVM selecciona el que produce el mayor margen o, en otras palabras, el que generaliza bien las diferencias entre los dos autores. En la Figura 3.3, tres puntos tienen una distancia M al borde, los puntos H59, M38 y M40. Al definir este límite de clase, el criterio subyacente es maximizar el margen y reducir el riesgo de devolver una solución sobreajustada, con respecto a la muestra de entrenamiento.

Después de determinar el límite de clase y S, el conjunto de vectores de soporte, SVM puede clasificar nuevas instancias. Para asignar el texto en disputa a uno de los dos autores (o categorías), el SVM calcula su distancia al borde. Cuando esta distancia es positiva, el texto se asigna al primer escritor, en caso contrario al segundo.

La popularidad y la eficacia del clasificador SVM están relacionadas con su capacidad para definir un límite de clase no lineal mediante la aplicación de funciones del kernel [48]. La función kernel es una generalización del producto interno y su forma más simple se muestra en la Ecuación 3.18 correspondiente a un núcleo lineal. Como se puede ver, este primer núcleo es el producto interno clásico [48].

$$K_{linear}(A, B) = \langle A, B \rangle = \sum_{i}^{m} a_i \cdot b_i \tag{3.18}$$

Para ser precisos, solo cuando se aplica un kernel no lineal, el clasificador podría llamarse *máquinas de vectores de soporte* (SVM) [48]. En lugar de limitarse a un límite de clase lineal, se pueden aplicar funciones kernel más complejas, por ejemplo, un kernel polinomial o un núcleo radial. La selección de una función kernel no lineal permite que el clasificador SVM dibuje un límite de clase no lineal. Como ejemplo, la Figura 3.4 muestra un borde que separa perfectamente los artículos de Madison y Hamilton cuando se usa la frecuencia relativa de *the* y *by*. El límite de clase debe ser no lineal en este contexto y se aplicó una función kernel polinomial.

En [92, 93, 99] proponen un modelo basado en SVM para la tarea de atribución de autoría de domino cruzado, el cuál también es aplicable a verificación de autoría. Este método se basa

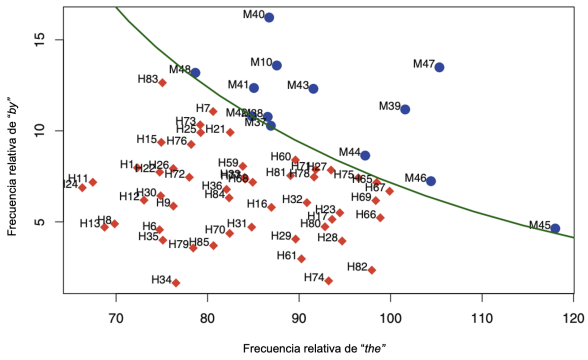

Figura 3.4: Un límite de clase no lineal entre los artículos de Hamilton y Madison. Imagen tomada de [48].

en características de n-gramas de caracteres y un clasificador de máquina de vectores de soporte (SVM). En primer lugar, todos los n-gramas de caracteres que aparecen al menos cinco veces en los textos de entrenamiento (autoría conocida) de un problema de atribución se extraen y utilizan como características para representar tanto los textos de entrenamiento como los de prueba. Luego, se entrena un SVM con kernel lineal basado en los textos de entrenamiento y se puede usar para predecir el autor más probable de los textos de prueba. Este modelo simple puede ser muy efectivo en condiciones de dominio cruzado dado que el número de características se define adecuadamente para cada problema de atribución específico [103]. Este método es utilizado como baseline para la tare de identificación de autoría del PAN2018 [56].

3.2.4. Métodos basados en aprendizaje profundo

Recientemente, los modelos de aprendizaje profundo se han utilizado ampliamente para aprender representaciones de texto, incluidas las redes neuronales convolucionales (CNN) y las redes neuronales recurrentes (RNN). Como CNN y RNN priorizan la localidad y la secuencialidad, estos modelos de aprendizaje profundo pueden capturar bien la información semántica y sintáctica en secuencias de palabras consecutivas locales [115].

3.2.4.1. Redes neuronales convolucionales (CNNs)

Las redes convolucionales, son un tipo especializado de red neuronal para procesar datos que tienen una topología conocida similar a una cuadrícula. Los ejemplos incluyen datos de series temporales, que se pueden considerar como una cuadrícula 1D que toma muestras a intervalos de tiempo regulares, y datos de imágenes, que se pueden considerar como una cuadrícula 2D de píxeles. Las redes convolucionales han tenido un gran éxito en aplicaciones prácticas [36], como detección y reconocimiento de objetos [34], segmentación semántica de imágenes [71], reconocimiento de expresiones faciales [37], procesamiento de texto, análisis de sentimiento [57], reconocimiento de voz y procesamiento del habla [1].

Las CNNs utilizan capas con filtros convolucionales que se aplican a características locales [57]. Originalmente inventados para la visión por computadora, los modelos CNN han demostrado ser efectivos en PLN y han logrado excelentes resultados en análisis semántico [116], recuperación de consultas de búsqueda, análisis de autoría y otras tareas tradicionales de PNL.

La arquitectura que se muestra en la Figura 3.5, es una ligera variante de la arquitectura CNN presentada en [24]. Sea $x_i \in \mathbb{R}^k$ el vector de palabras k-dimensional correspondiente a la i-ésima palabra en la oración. Una oración de longitud n se representa como:

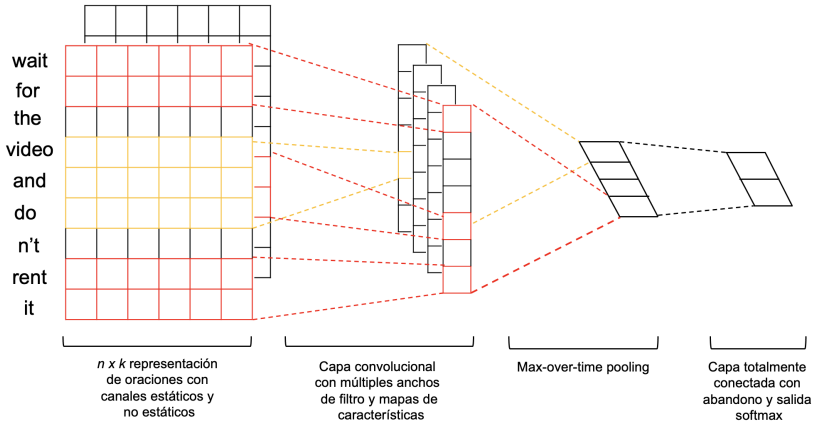

Figura 3.5: Modelo de arquitectura con dos canales para una oración. [116].

$$x_{1:n} = x_1 \oplus x_2 \oplus ... \oplus x_n, \tag{3.19}$$

Donde $\oplus$ es el operador de concatenación. En general, sea $x_{i:i+j}$ la concatenación de palabras $x_i, x_{i+1}, ..., x_{i+j}$. Una operación de convolución implica un filtro $w \in \mathbb{R}^{hk}$, que se aplica a una ventana de h palabras para producir una nueva característica. Por ejemplo, una característica c_i se genera a partir de una ventana de palabras $x_{i:i+h-1}$ mediante la Ecuación 3.20.

$$c_i = f(w \cdot x_{i:i+h-1} + b) \tag{3.20}$$

Aquí $b \in \mathbb{R}$ es un término de sesgo y f es una función no lineal como la tangente hiperbólica. Este filtro se aplica a cada ventana posible de palabras en la oración $x_{1:h}, x_{2:h+1}, ..., xn - h + 1 : n$ para producir un *mapa de características.*

$$c = [c_1, c_2, ..., c_{n-h+1}] \tag{3.21}$$

Con $c \in \mathbb{R}^{n-h+1}$. Luego se aplica una operación de agrupación de max-over-time pooling [24] sobre el mapa de características y se toma el valor máximo $\hat{c} = max\{c\}$ como la característica correspondiente a este filtro en particular. La idea es capturar la característica más importante, una con el valor más alto, para cada mapa de características. Este esquema de agrupación, naturalmente trata con oraciones de longitud variable.

Este es el proceso mediante el cual se extrae una característica de un filtro. El modelo utiliza múltiples filtros (con diferentes tamaños de ventana) para obtener múltiples características. Estas características forman la penúltima capa y se pasan a una capa de clasificación cuyo resultado es la distribución de probabilidad sobre las etiquetas [116].

Schaetti en [94] propone un enfoque como solución a la tarea de cambio de estilo (que en principio la tarea de cambio de estilo se puede modelar como una serie de casos de verificación de autoría) para el PAN2018 [56] el cuál diseña una red neuronal convolucional (CNN) basada en caracteres. Cada documento se representa como un vector de tamaño fijo de 12 000

caracteres consecutivos que se introducen en la red, es decir, en una capa de embedding que reduce la dimensión a 50 y captura las similitudes de contexto de los caracteres que ocurren en un espacio multidimensional. Posteriormente, la segunda capa se compone de tres capas convolucionales diferentes con 25 filtros cada una para capturar los patrones más expresivos de 2-4 caracteres consecutivos de 2 gramas. Después de utilizar una capa de agrupación máxima para cada una de las capas convolucionales, finalmente se usa una capa lineal binaria para predecir la existencia de cambios de estilo. En cuanto a la precisión, este modelo llega último de los cinco participantes en el PAN2018 [56] pero segundo en términos de tiempo de ejecución.

En [83] usaron un modelo de lenguaje T5 como una capa base de embedding, seguida de una red neuronal convolucional y un mecanismo de atención para extraer características locales y contextuales de los textos. Como resultado del estudio de múltiples modelos de lenguaje y arquitecturas de aprendizaje profundo, obtuvieron una exactitud de 91.79 % en su conjunto de datos de prueba. Sin embargo, en el conjunto de datos oficial del PAN 2022, obtuvieron un promedio de 58.7 %.

3.2.4.2. Redes neuronales recurrentes (RNNs)

La función principal de los modelos neuronales es representar el texto de longitud variable como un vector de longitud fija. Estos modelos generalmente consisten en una capa de proyección que asigna palabras, unidades de subpalabras o n-gramas a representaciones vectoriales (a menudo pre-entrenadas con métodos no supervisados), y luego las combina con las diferentes arquitecturas de redes neuronales.

Hay varios tipos de modelos para modelar texto, como el modelo Neural Bag-of-Words (NBOW), la red neuronal recurrente (RNN), la red neuronal recursiva (RecNN) y red neuronal convolucional (CNN). Estos modelos toman como entrada los embeddings de palabras en la secuencia del texto y resumen su significado con una representación vectorial de longitud fija. Entre ellas, las redes neuronales recurrentes (RNN) son una de las arquitecturas más populares utilizadas en problemas de PLN debido a que su estructura recurrente es muy adecuada para procesar texto de longitud variable.

Una red neuronal recurrente (RNN) [68] es capaz de procesar una secuencia de longitud arbitraria aplicando recursivamente una función de transición a su vector de estado oculto interno h_t de la secuencia de entrada. La activación del estado oculto h_t en el paso de tiempo t se calcula como una función f del símbolo de entrada actual x_t y el estado oculto anterior h_{t-1}

$$h_t = \begin{cases} 0 & t = 0 \\ f(h_{t-1}, x_t) & \text{otherwise} \end{cases} \tag{3.22}$$

Es común usar la función de transición de estado a estado f como la composición de una no linealidad por elementos con una transformación afín de x_t y h_{t-1}. Tradicionalmente, una estrategia simple para el modelado de secuencias es mapear la secuencia de entrada a un vector de tamaño fijo usando una RNN y luego alimentar el vector a una capa de clasificación u otras tareas [22].

En [46] proponen el uso de RNN para solucionar la tarea de cambio de estilo de escritura (identificar posiciones de texto dentro de un documento de varios autores determinado en el que el autor cambia). El modelo está inspirado en las dos arquitecturas de redes neuronales exitosas en verificación de autoría y clasificación de documentos. La idea principal de este enfoque es confiar únicamente en la estructura gramatical utilizada por los autores para detectar cambios de estilo, es decir, no se utilizan otras características léxicas como n-gramas de caracteres o

palabras. Se obtiene la estructura jerárquica de cada oración de un documento dado, que luego se recorre y linealiza. Al hacerlo, todo el documento se representa como un orden consecutivo de características del árbol de análisis. Para usar esta estructura de árbol con las siguientes LSTM en el modelo Figura 3.6, se necesita preservar la secuencia de palabras de la oración. Se define la función de árbol de análisis (PTF) de cada palabra en una oración como una ruta que comienza desde la raíz hasta la hoja (palabra) correspondiente de su árbol de análisis. La ruta es un conjunto de todas las reglas de la forma $padre \rightarrow hijo_1...hijo_n$ desde la raíz hasta la hoja (palabra). Con esto se construye una segunda RNN Figura 3.6, cuya entrada es la representación de características del árbol de análisis del orden inverso de las oraciones del documento. Finalmente, se tiene un paso de fusión donde se calculan múltiples métricas de similitud para estimar la diferencia entre las representaciones de red de orden inverso y original, donde una capa softmax final produce la predicción de cambio de estilo.

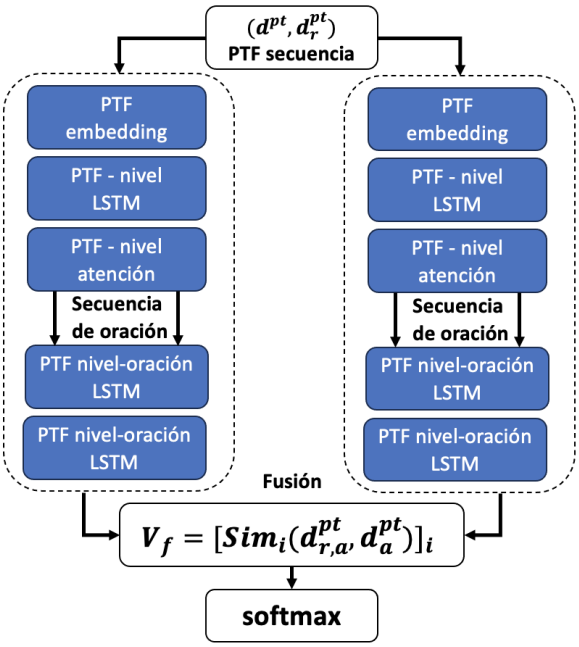

Figura 3.6: Arquitectura de red de atención jerárquica paralela. [46].

A diferencia de las redes neuronales recurrentes convencionales que usan secuencias de caracteres o palabras para aprender el modelo de lenguaje subyacente de los documentos, este modelo se enfoca en la estructura jerárquica del lenguaje y observa las características del árbol de análisis de una oración usando un analizador estadístico previamente entrenado. Además, el modelo es independiente de las posiciones de cambio de estilo aunque se dan durante la fase de entrenamiento. La razón es tener un enfoque más aplicable a los problemas del mundo real donde dicha información no está disponible. Los resultados de PAN 2018 muestran que logra un 82 % de precisión [56].

En [49] se introdujo una red neuronal recurrente sintáctica para codificar los patrones sintácticos de un documento en una estructura jerárquica. Este método primero aprende la representación sintáctica de la oración de la secuencia de etiquetas POS. Los resultados experimentales con el conjunto de datos del PAN 2012 para la tarea de atribución de autoría muestran que la red neuronal recurrente sintáctica (trabajando sobre vectores de etiquetas POS) supera al modelo léxico (trabajando sobre vectores de palabras) con una arquitectura idéntica por aproximadamente 14 % en términos de exactitud.

Desafortunadamente, un problema con las RNN's con funciones de transición de esta forma es que durante el entrenamiento, los componentes del vector gradiente pueden crecer o decaer exponencialmente en secuencias largas [43]. Este problema con los gradientes que explotan o se desvanecen dificulta que el modelo RNN aprenda correlaciones de larga distancia en una secuencia.

En [44] proponen la red de memoria a largo plazo (LSTM por sus siglas en inglés) para abordar específicamente este problema del aprendizaje de las dependencias a largo plazo. LSTM mantiene una celda de memoria separada en su interior que actualiza y expone su contenido solo cuando se considera necesario. Se definen las unidades LSTM en cada paso de tiempo t como una colección de vectores en $\mathbb{R}^d$: una puerta de entrada i_t, una puerta de olvido f_t, una puerta de salida o_t, una celda de memoria c_t y un estado oculto h_t. El número de unidades LSTM está representado por d. Las entradas de los vectores de activación i_t, f_t y o_t están en $[0, 1]$.

El lenguaje natural exhibe propiedades sintácticas que combinan naturalmente palabras con frases. En [105] presentan el modelo Tree-LSTM, una generalización de LSTM para topologías de red con estructura de árbol. Los TreeLSTM superan a todos los sistemas existentes y las sólidas líneas base de LSTM en dos tareas: predecir la relación semántica de dos oraciones y la clasificación de sentimientos.

3.2.4.3. Redes neuronales transformers

Los *transformers* son presentados por Google en el documento "Attention Is All You Need [109]". Los *transformers* son redes neuronales de secuencia a secuencia, diseñadas para manejar datos secuenciales como reconocimiento de voz, transformación de texto a voz, resumen y traducción con dependencias a largo plazo. Los modelos de *transformer* consisten en un codificador que toma una secuencia de entrada y la asigna a un espacio dimensional superior (vector n dimensional) y un decodificador que convierte la secuencia codificada en una secuencia de salida. Al entrenar el modelo *transformer*, el método de atención entra en juego en cada paso que decide qué otras partes de la secuencia son importantes y les asigna pesos en función de su importancia. Los modelos de *transformers* utilizan capas de avance y capas de atención apiladas unas sobre otras, lo que les permite manejar entradas de tamaño variable. Cada capa puede procesar en paralelo. Los *transformers* han demostrado un rendimiento notable en varias tareas de PLN, como la traducción automática [109], la generación de documentos [69] y el análisis sintáctico [59]. Transferir el aprendizaje es otro enfoque que utiliza el conocimiento aprendido de un conjunto de datos y lo aplica a varios otros problemas. Ha demostrado ser muy eficaz para proporcionar una buena precisión con cantidades muy pequeñas de datos etiquetados [47].

Las representaciones de codificador bidireccional de *transformers* (BERT) [26] es un modelo poderoso de Google AI entrenado en una gran cantidad de datos sin etiquetar. A diferencia de otras estructuras de redes neuronales para PLN, BERT es una representación de codificador bidireccional de transformers que incluye completamente las características contextuales bidireccionales dentro del modelo. Las representaciones bidireccionales profundas se entrenan incorporando contextos izquierdo y derecho en todas las capas. Estas representaciones son aprendidas por dos tareas no supervisadas en una gran cantidad de datos sin etiquetar. La primera tarea se llama LM enmascarado, que enmascara un porcentaje de los tokens de entrada al azar y luego predice esos tokens enmascarados. La segunda tarea se llama predicción de la siguiente oración, que es una tarea de predicción binaria [47]. Está pre-entrenado por un par de dos oraciones, A y B, donde el 50 % de las veces B es la siguiente oración real que sigue

a A, y el 50 % de las veces es una oración aleatoria del cuerpo del texto. Utiliza entrenamiento previo y ajuste para crear modelos de última generación para diferentes tareas de PLN, como sistemas de respuesta a preguntas [70], análisis de sentimientos [104], generación de texto y resumen automático [114]. En la fase de pre-entrenamiento, el modelo se entrena en una gran cantidad de texto sin etiquetar y para el ajuste, el modelo se inicializa primero con los parámetros pre-entrenados que se ajustan con una pequeña cantidad de datos etiquetados de la tarea posterior. Cada tarea posterior tiene modelos ajustados separados [47]. En [4] se examinó el uso de modelos de lenguaje previamente entrenados para la atribución de autoría de dominio cruzado. Sus resultados demostraron que los modelos de lenguaje previamente entrenados como BERT funcionan bien y su rendimiento se mantiene estable en diferentes escenarios. En este enfoque, la clasificación se realiza ajustando el modelo BERT en datos específicos del dominio. El modelo propuesto consta de dos partes, el modelo de lenguaje (LM) y el clasificador multi-cabezal (MHC). La capa DEMUX en la parte MHC funciona como un demultiplexor, su estado lo define el selector. Durante la fase de entrenamiento, el selector lo define el autor del texto de entrada y durante el cálculo del vector de normalización o fase de prueba, la entrada de DEMUX está conectada a todas sus salidas.

En [60] probaron varios modelos de codificación basados en *transformers* para la tarea de verificación de autoría del PAN2022 [101], usando enfoques de codificación cruzada (Cross-Encoder) y codificación bi-direccional (Bi-Encoder). Los modelos *transformers*, como BERT, capturan el contexto de las palabras en un texto utilizando atención y auto-atención. Esto, en comparación a otros métodos, les permite capturar mejor las relaciones y dependencias entre las palabras, lo cual es fundamental en la verificación de autoría, donde los estilos de escritura y las elecciones léxicas pueden variar entre diferentes autores. Además, se pre-entrenan en grandes corpus de texto sin etiquetas, lo que les permite aprender representaciones generales del lenguaje. Esto es beneficioso, ya que los modelos pueden aprovechar el conocimiento previo adquirido durante el pre-entrenamiento y adaptarlo a tareas específicas de verificación de autoría.

3.2.5. Métodos basados en grafos

Los modelos de aprendizaje profundo CNN y RNN priorizan la localidad y la secuencialidad [6], estos modelos pueden capturar bien la información semántica y sintáctica en secuencias locales de palabras consecutivas, pero pueden ignorar la co-ocurrencia global de palabras en un corpus que lleva semántica no consecutiva y de larga distancia [88]. Una dirección de investigación llamada redes neuronales gráficas o embeddings de grafos ha atraído una gran atención [6, 17]. Las redes neuronales de gráficos han sido eficaces en tareas que se cree que tienen una estructura relacional rica y pueden preservar la información de la estructura global de un gráfico en embeddings de grafos.

Han habido una serie de estudios que aplican redes neuronales convolucionales (convolución en cuadrícula regular, por ejemplo, secuencia) para clasificación. Sin embargo, solo un número limitado de estudios han explorado las redes neuronales convolucionales de grafos más flexibles (convolución en no cuadrícula, por ejemplo, grafo arbitrario) para las tareas de análisis de autoría. En [115] proponen utilizar redes convolucionales de grafos para la clasificación de textos. Construyen un único gráfico de texto para un corpus basado en la co-ocurrencia de palabras y las relaciones entre palabras, luego entrenan una red convolucional de grafos de texto (GCN) para el corpus. La GCN de texto se inicializa con una representación única para la palabra y el documento, luego aprende conjuntamente los embeddings de las palabras y los documentos, según lo supervisado por las etiquetas de clase conocidas para los documentos. Los resultados

experimentales de este modelo en múltiples conjuntos de datos de referencia demuestran que un GCN de texto estándar sin embeddings de palabras externas o conocimiento supera a los métodos de última generación para la clasificación de texto. Por otro lado, La GCN también aprende embeddings de palabras y documentos. Además, los resultados experimentales muestran que la mejora de GCN sobre los métodos de comparación de última generación se vuelve más prominente a medida que se reduce el porcentaje de datos de entrenamiento, lo que sugiere la solidez de la GCN a menos datos de entrenamiento en la clasificación de texto. Así este método puede ser aplicado para análisis de autoría.

En [29] propusieron un enfoque que consiste en una representación basada en grafos para representar los textos, que sirvió como entrada a una red siamesa. La red de extracción de características consistió en capas de embeddings de nodos para obtener representaciones vectoriales para cada nodo en el grafo, así como una global pooling. También incorporaron características estilométricas, combinándolas con los componentes del grafo en un conjunto.

En [110] presentan un enfoque para la tarea de verificación de autoría del PAN 2013 [52] que utiliza características léxicas, sintácticas y basadas en grafos para construir un modelo de representación de autores de documentos. En particular, las características extraídas de la representación gráfica se obtuvieron mediante la herramienta de minería SubDue [110]. Como modelo de clasificación se utilizan máquinas de vectores de soporte (SVM). Los resultados generales han clasificado el enfoque en el quinto lugar de alrededor de 17 equipos con una precisión del 65 %. Por otro lado en [20] proponen distintos tipos de grafos para representar textos en varias tareas de análisis de autoría. Para la tarea de Verificación de Autoría se propone un grafo de co-ocurrencia directo [19]. La diferencia entre este grafo y el gráfico utilizado para extraer características [110] es que se utiliza un esquema ponderado de borde basado en las relaciones naturales de las palabras en un contexto que considera la similitud entre los grafos. Formalmente, el grafo propuesta se representa por $G = (V, E, LV, LE)$, donde:

- $V = v_1, ..., v_n$ es un conjunto finito de vértices que representan las palabras contenidas en un o muchos textos.
- $E \subseteq V \times V$ es el conjunto finito de aristas las cuales representan que dos vértices están conectados si sus unidades léxicas correspondientes co-ocurren dentro de una ventana.
- L_v es el conjunto de etiquetas de V, donde $L_v = etq : eta \in words$
- L_E es el conjunto de etiquetas de E, las cuales consisten del número de veces que dos vértices co-ocurren en una ventana de texto de dos palabras.

3.3. Resumen

En esta capítulo se resumen las técnicas y métodos que han sido desarrollados hasta el momento para resolver la tarea de verificación de autoría y algunos otros ejemplos que también se han presentado que si bien no fueron aplicados para solucionar la tarea de verificación de autoría, se han aplicado al análisis de autoría o para representar las características estilísticas de los autores en los textos, que en escénica es lo que se busca cuando se requiere obtener la representación de un texto del que se quiere verificar su autoría. También se describen modelos de aprendizaje automático y aprendizaje profundo que son muy aplicados en general para tareas de clasificación pero en específico en este caso para extraer las características requeridas de los textos y posteriormente verificar la autoría de los documentos. Además, Se describen enfoques que son relativamente sencillos y rápidos de implementar que se utilizan como referencia para comparar el rendimiento de modelos más complejos.

Capítulo 4

Metodología grafo de texto heterogéneo

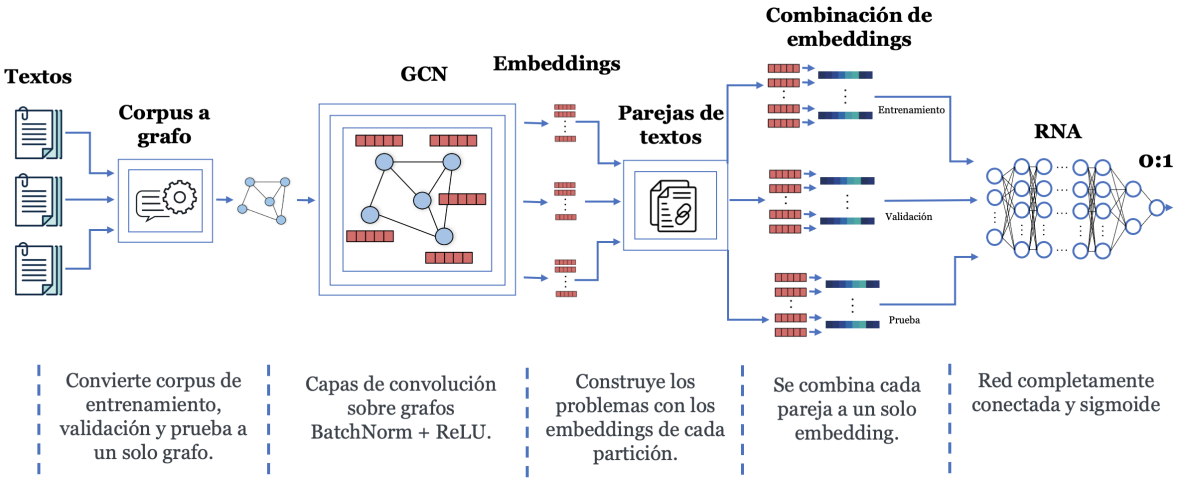

Figura 4.1: Metodología grafo de texto heterogéneo.

En este capítulo se describe la arquitectura del modelo propuesto en este trabajo de tesis Figura 4.1 con el objetivo de resolver la tarea de verificación de autoría de discurso cruzado. Como se ha presentado en capítulos anteriores, existen diversas formas de abordar la tarea y los principales problemas que se enfrentan. Desde la adquisición de un corpus apropiado, la representación de las características de los textos, el método de aprendizaje automático o aprendizaje profundo, etc. Una parte importante en la que está enfocada el trabajo de tesis es en el estudio de la representación de características estilísticas de los textos utilizando grafos, en este capítulo se explica la representación basada en grafos utilizada y cada sección de la arquitectura del modelo.

4.1. Modelo de texto como grafo.

La propuesta que se presenta en este trabajo es aprovechar las bondades que ofrecen los grafos en los textos y las redes convolucionales para extraer características. Con esto, se construye un grafo de texto grande y heterogéneo como el propuesto en [115] que contiene nodos

de palabras y nodos de documentos para que la co-ocurrencia global de palabras se pueda modelar explícitamente y la convolución del gráfico se pueda adaptar fácilmente.

La construcción del grafo heterogéneo, busca modelar todo un corpus como un solo grafo. Utilizando nodos de palabras y de documentos y con esto obtener una representación de documento completa, también capturar la relación entre palabras y etiquetas POS en los textos. A continuación se ejemplifica el proceso para crear el grafo, tomando el siguiente texto extraído del corpus PAN 2022 como ejemplo.

<nl>I am a Second year <course> student at <university>. I am interested.

Primero se realiza un preprocesamiento al texto que consta de los siguientes pasos:

- Se sustituyen non-ASCII caracteres por su equivalente en ASCII.
- Se pasa todo a minúsculas.
- Se realiza tokenización.
- Se obtienen las etiquetas POS de cada token.

Para obtener las etiquetas POS, se consideran las etiquetas PENN-Treebank POS [74]. Estas etiquetas son obtenidas con la ayuda de la librería NLTK [1] de Python. Dos etiquetas adicionales son definidas $PUNCT para marcar las puntuaciones y $OTHER para marcar alguna otra palabra que la librería NLTK no pueda identificar. En total son consideradas 38 etiquetas. Después de los procesos anteriores se obtiene la siguiente lista de tuplas compuestas por el token y su respectiva etiqueta POS. Con estas tuplas se definirán algunos de los nodos del grafo:

**[('<nl>','NN'),('i','PRP'),('am','VBP'),('a','DT'),('second','JJ'),('year','NN'),
('<course>','FW'),('student','NN'),('at','IN'),('<university>','NNP'),
('.','$PUNCT'),('i','PRP'),('am','VBP'),('interested','JJ'),('.','$PUNCT')]**

El número de nodos, representado por $|V|$, en el grafo de texto es el número de documentos (tamaño del corpus) más el número de palabras únicas (tamaño del vocabulario) o etiquetas POS en el corpus. Para definir los nodos correspondientes a las palabras o etiquetas POS se define un conjunto de etiquetas POS denotado como REDUCE_LABELS. Con esto, la estructura del grafo puede cambiar y también la información extraída del texto. Para definir formalmente la lista de nodos de palabras o etiquetas POS, sea P la lista de tuplas ya definida, $l(P)$ el número de elementos en la lista, y $P[i]$ el iésimo elemento en la lista. Por cada $P[i] = (token, POS)$ en P, se puede definir 4.1.

$$M[i] = \begin{cases} \text{token} & Si \quad \text{POS} \quad \notin \quad \text{REDUCE\_LABELS} \\ \text{POS} & Si \quad \text{POS} \quad \in \quad \text{REDUCE\_LABELS} \end{cases} \tag{4.1}$$

Donde M es la lista enmascarada de tokens o etiquetas POS como se explicó anteriormente. Con esta construcción, se definen todas las tuplas con una etiqueta específica en el conjunto de REDUCE_LABELS como un solo nodo. En los experimentos, se evalúa el modelo utilizando grafos generados con diferentes conjuntos de REDUCE_LABELS. Por ahora, se denota como *short graph* al grafo generado usando el conjunto de todas las posibles etiquetas POS

[1] `https://www.nltk.org/`

como REDUCE_LABELS, *full graph* al grafo generado usando REDUCE_LABELS = ∅ y se denomina *med graph* al grafo generado usando el conjunto de REDUCE_LABELS de la Tabla 4.1. Se consideran solo la información de la etiqueta POS por dos razones: Al limitar las categorías gramaticales utilizadas, se reduce el ruido y la complejidad en el texto. Algunas categorías gramaticales, como los artículos o las preposiciones, tienden a tener menos información relevante para el análisis en comparación con otras categorías como los sustantivos o los verbos. Al excluir categorías gramaticales menos informativas, se puede simplificar el texto y centrarse en las partes más relevantes para el análisis. Existen referencias donde un modelo de aprendizaje profundo que entrena sobre *embeddings* de POS en lugar de *embeddings* de palabras obtiene mejores resultados para una tarea de análisis de autoría [49]. Además, el uso de *embeddings* de POS de baja dimensión para representar nodos en lugar de *embeddings* de palabras de mayor dimensión nos permite reducir el costo computacional del modelo. Los resultados obtenidos con estas decisiones son comparables con modelos basados en *embeddings* de palabras, como podemos ver en [55].

REDUCE_LABELS = [	'JJ',	'JJR',	'JJS',		#Adjectives
	'NN',	'NNS',	'NNP',	'NNPS',	#Nouns
	'RB',	'RBR',	'RBS',		#Adverbs
	'VB',	'VBD',	'VBG',		#Verbs
	'VBN',	'VBP',	'VBZ',		#Verbs
	'CD',				#Cardinal numbers
	'FW',				#Foreign words
	'LS',				#List item marker
	'SYM',				#Symbols
	'$OTHER',				# Others]

Tabla 4.1: Conjunto de REDUCE_LABELS usado para el grafo *med graph*.

Entonces, el tamaño del grafo depende del conjunto REDUCE_LABELS utilizado en la construcción y el número de documentos. Continuando con el ejemplo, el *short graph*, *med graph* y *full graph* se muestran en las Figuras 4.2, 4.3 y 4.4 respectivamente; para hacer menos complejas las imágenes no se incluyen los pesos de las aristas.

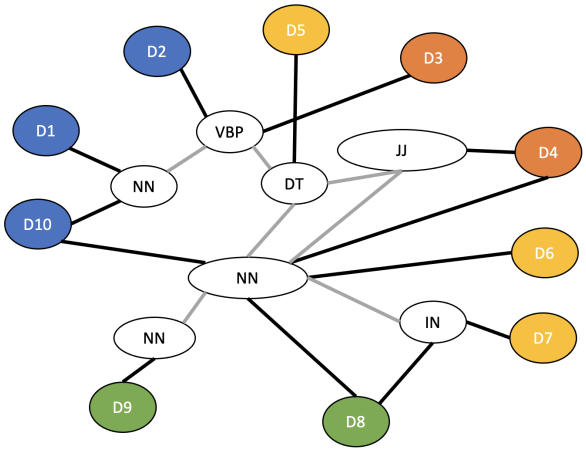

Figura 4.2: Grafo Full.

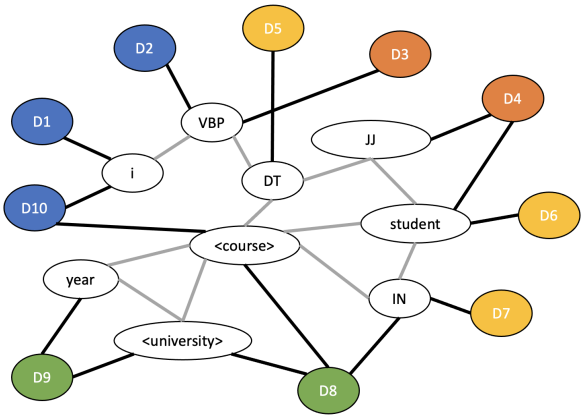

Figura 4.3: Grafo Med.

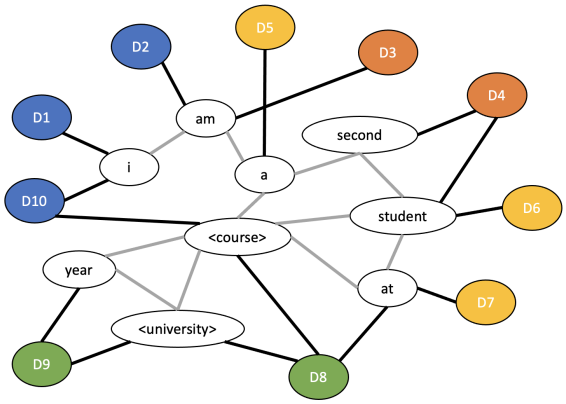

Figura 4.4: Grafo short.

Formalmente el grafo se define como $G = (V, E)$, donde $V(|V| = n)$ y E son conjuntos de nodos y aristas, respectivamente. Sea $X \in \mathbb{R}^{n \times m}$ una matriz que contiene todos los n nodos con sus características, donde m es la dimensión de los vectores de características, cada fila $x_v \in \mathbb{R}^m$ es el vector de características para v. Se define una matriz de adyacencia A de G y su matriz de grado D, donde $D_{ii} = \sum_j A_{ij}$. Los elementos de la diagonal de A se establecen en 1 debido a los bucles.

Ahora se establece la matriz de características $X = I$ como una matriz de identidad, lo que significa que cada palabra, etiqueta POS o documento se representa como un vector one-hot como entrada a la GCN de texto. Se establecen aristas entre nodos basados en la ocurrencia de palabras en documentos (aristas de documento-palabra) y la co-ocurrencia de palabras en todo el corpus (aristas de palabra-palabra). El peso de la arista entre un nodo de documento y un nodo de palabra es la Frecuencia de Término × Frecuencia Inversa de Documento (TF-IDF) de la palabra en el documento, donde la frecuencia de término es el número de veces que aparece la palabra en el documento, la frecuencia inversa de documento es la fracción inversa escalada logarítmicamente del número de documentos que contienen la palabra. En [115] se demuestra que usar el peso TF-IDF es mejor que usar solo la frecuencia de término. Para utilizar la información global de co-ocurrencia de palabras, se utiliza una ventana deslizante de tamaño fijo en todos los documentos del corpus para recopilar estadísticas de coocurrencia. Se emplea información mutua puntual (PMI), una medida para asociaciones de palabras, para calcular

pesos entre dos nodos de palabras. También en [115] se descubre que usar PMI logra mejores resultados que usar el conteo de co-ocurrencia de palabras en experimentos. Formalmente, los pesos de las aristas entre un nodo i y un nodo j se define en 4.2.

$$A_{ij} = \begin{cases} PMI(i,j) & i,j \quad \text{son palabras,} \quad PMI(i,j) > 0 \\ TF - IDF_{ij} & i \quad \text{es documento,} \quad j \quad \text{es palabra} \\ 1 & i = j \\ 0 & \text{todos los demás casos} \end{cases} \tag{4.2}$$

El valor PMI de un par de palabras i, j es calculada de la siguiente forma.

$$PMI(i,j) = log\frac{p(i,j)}{p(i)p(j)} \tag{4.3}$$

$$p(i,j) = \frac{\#W(i,j)}{\#W} \tag{4.4}$$

$$p(i) = \frac{\#W(i)}{\#W} \tag{4.5}$$

Donde $\#W(i)$ es el número de ventanas deslizantes en el corpus que contiene la palabra i, $\#W(i,j)$ es el número de ventanas deslizantes que contienen ambas palabras i y j, y $\#W$ es el número total de ventanas deslizantes en el corpus. Cabe mencionar que un valor positivo de PMI implica una alta correlación semántica de palabras en un corpus [115], mientras que un valor negativo de PMI indica poca o nula correlación semántica en el corpus [115]. Es por ello que, en esta implementación solo se consideran aristas entre pares de palabras con un valor de PMI positivo. Para que se comprenda mejor, a continuación se ejemplifica el método para obtener los pesos de las aristas de los grafos. Tomando en cuenta los siguientes textos extraídos de distintos documentos del corpus, para este ejemplo cada oración representa un documento.

- Documento 1: *I am a hard-working student*
- Documento 2: *he reports stress about university*
- Documento 3: *John is an 18-year-old male university student*

Primero se calculan los pesos entre los nodos de documento y de palabras o etiquetas POS, para esto se calcula la frecuencia de término - frecuencia inversa de documento (TF-IDF). En este caso se hará para la palabra "*student*".

Documento	$TF(t,d)$	$IDF(t)$	$TF - IDF(t,d)$
1	1/5	$log(3/2)$	0.035
2	0/5	$log(3/2)$	0
3	1/7	$log(3/2)$	0.028

Tabla 4.2: Ejemplo de cálculo del TF-IDF.

Donde:

$$TF(t,d) = \frac{\text{Veces que } t \text{ aparece en } d}{\text{total de términos en } d} \tag{4.6}$$

$$IDF(t) = log\frac{\text{Total de documentos}}{|d \in D : t \in d|} \tag{4.7}$$

$$TF - IDF(t,d) = TF(t,d) \times IDF(t) \tag{4.8}$$

Ahora para calcular el valor PMI (los pesos de las aristas entre palabras) del par de palabras (university,student) se calcula de la siguiente manera utilizando las Ecuaciones 4.3, 4.4, 4.5 y suponiendo un tamaño de ventana de 3. Para este caso se consideran los siguientes documentos:

- Documento 1: *I am a* | *hard-working student*
- Documento 2: *he reports stress* | *about university student*
- Documento 3: *John is an* | *male university student*

En los diferentes textos se observa que las palabras están encerradas en cajas, estas cajas representan las ventanas que hay en cada documento, es decir, como se definió un tamaño de ventana de 3, cada caja debe de contener 3 palabras excepto las últimas palabras del documento que puede variar por las palabras que resten del documento. Esto ayuda a comprender mejor el cálculo del valor PMI.

$$p(i,j) = \frac{\#W(i,j)}{\#W} = \frac{2}{6} = 0.33 \tag{4.9}$$

$$p(i) = \frac{\#W(i)}{\#W} = \frac{2}{6} = 0.33 \tag{4.10}$$

$$p(j) = \frac{\#W(j)}{\#W} = \frac{3}{6} = 0.5 \tag{4.11}$$

$$PMI(i,j) = log\frac{p(i,j)}{p(i)p(j)} = log\frac{0.33}{0.33 \times 0.5} = 0.301 \tag{4.12}$$

Donde:

- $\#W(i)$: ventanas en el corpus que contienen la palabra i.
- $\#W(j)$: ventanas en el corpus que contienen la palabra j.
- $\#W(i,j)$: ventanas que contienen ambas palabras i y j.
- $\#W$: total de ventanas que contiene el corpus.

Ya obtenidos los pesos de las aristas y los distintos nodos el grafo se ve de la siguiente forma 4.5.

En la Figura 4.5 los nodos de colores representan los nodos de documento y los colores son los distintos autores. Los nodos que están de color blanco son los nodos de palabras. En el grafo también se aprecian las aristas y el tipo de peso que les corresponde tanto para las aristas de documento-palabra como las aristas de palabra-palabra.

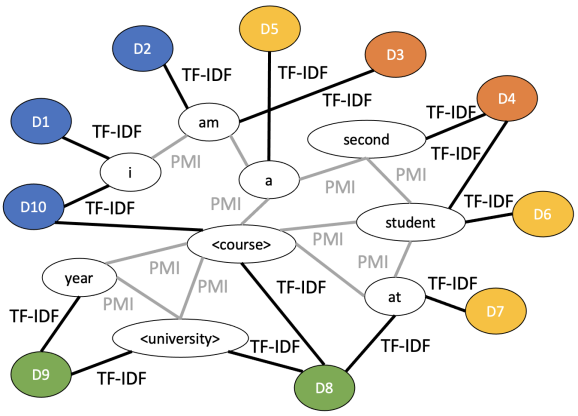

Figura 4.5: Grafo full con pesos en aristas.

4.2. Modelo de red neuronal convolucional basada en grafos (GCN).

Después de tener el grafo de texto construido, se introduce al componente de extracción de características, el cuál esta compuesto por una red convolucional basada en grafos y regresa embeddings de documentos; el objetivo es extraer características relevantes que puedan identificar el estilo del autor de la representación vectorial del documento por medio del grafo que modela al corpus.

Este enfoque de red neuronal convolucional basada en grafos (GCN por sus siglas en inglés) ha sido aplicada por varios autores [17], por ejemplo, para extraer entidades médicas de un texto [113], extraer hechos relacionales de texto sin formato a gran escala [111] y vincular texto en lenguaje natural con entidades en un grafo de conocimiento [112]. Estos autores han generalizado modelos de redes neuronales bien establecidos como CNN que se aplican a la estructura de cuadrícula regular (malla de 2 dimensiones o secuencia de 1 dimensión) para trabajar en grafos estructurados arbitrariamente [15, 25, 41, 58]. En primeros enfoques, Kipf y Welling presentaron un modelo de red neuronal de grafos simplificado, llamado red neuronal convolucional basada en grafos (graph convolutional networks en inglés), la cuál alcanzó los resultados de clasificación de los modelos del estado del arte en una serie de conjuntos de datos de grafos de referencia [58]. La GCN también se exploró para varias tareas de PLN, como el etiquetado de roles semánticos [73], clasificación de relaciones [67] y la traducción automática [5] donde la GCN se utiliza para codificar la estructura sintáctica de oraciones.

Una red neuronal convolucional basada en grafos [58] es una red neuronal multicapa que opera directamente en un grafo e induce embeddings de nodos en función de las propiedades de sus vecinos. Entonces en este enfoque la primera capa toma como entrada un grafo con un vector de características inicial en cada nodo; cada vector de nodo inicial que representa un texto completo tiene una dimensión m por su representación one-hot.

GCN puede capturar información solo sobre vecinos inmediatos con una capa de convolución. Cuando se apilan varias capas de GCN, se integra la información sobre vecinos más lejanos. Para una GCN de una capa, sea $L^{(1)}$ la nueva matriz de características del nodo k-dimensional, donde $L^{(1)} \in \mathbb{R}^{n \times k}$ se calcula con la Ecuación 4.13.

$$L^{(1)} = \rho(\tilde{A}XW_0) \tag{4.13}$$

Donde, X es la matriz que contiene todos los n nodos con sus características, $\tilde{A} = D^{-\frac{1}{2}}AD^{\frac{1}{2}}$ es la matriz de adyacencia simétrica normalizada y $W_0 \in \mathbb{R}^{m \times k}$ es una matriz de pesos. ρ es una función de activación, por ejemplo una ReLU $\rho(x) = max(0, x)$. Como se mencionó antes, se puede capturar información de vecinos más lejanos apilando múltiples capas de GCN. Entonces una GCN multicapa puede permitir el paso de mensajes entre nodos que están a múltiples pasos de distancia. Así, aunque no hay aristas directas de documento a documento en el grafo, la GCN permite el intercambio de información entre pares de documentos. La Ecuación 4.13 se puede re-escribir para una GCN multicapa 4.14.

$$L^{(j+1)} = \rho(\tilde{A}L^{j}W_j) \tag{4.14}$$

Donde j denota el número de capas y $L^{(0)} = X$.

La salida de cada capa de es la misma estructura de grafo con nuevos vectores de características en cada nodo; la dimensión de los vectores obtenidos se puede definir de la misma manera que se definen los canales utilizados en una capa convolucional tradicional. En esta arquitectura la dimensión del embedding es un hiperparámetro, así como el número de capas de convolución. El modelo general de la GCN de texto se ilustra esquemáticamente en la Figura 4.6.

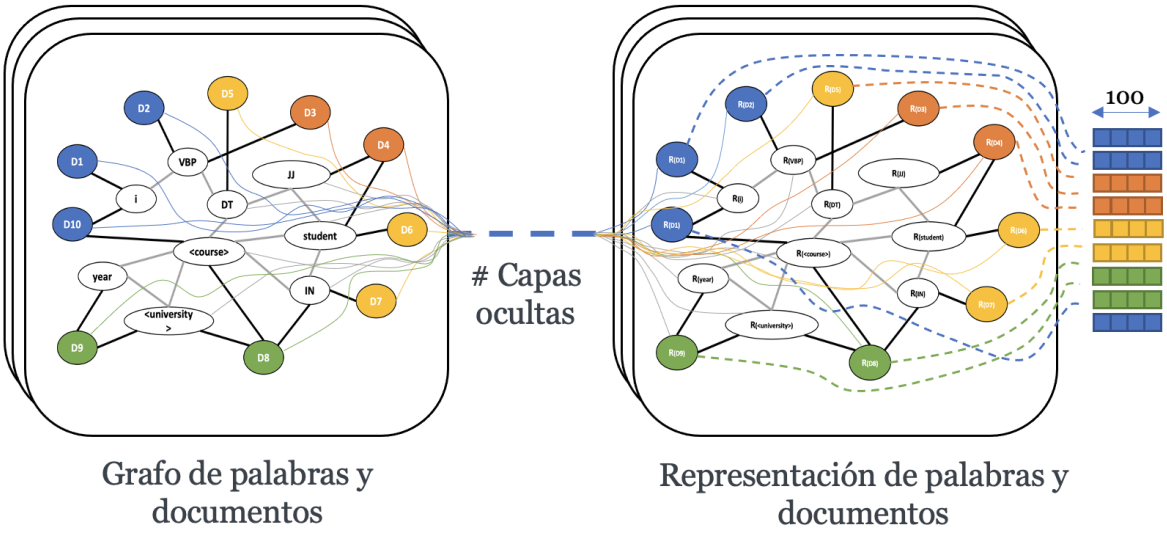

Figura 4.6: Red neuronal convolucional basada en grafos.

Con los embeddings de documentos generados, se procede a un paso de reducción. Simplemente se toman los embeddings de cada documento y se forman las parejas que correspondan a los casos de verificación de autoría de discurso cruzado. Para reducir ambos embbedings a uno solo se utilizan dos opciones, la primera concatenación y la segunda el valor absoluto de la diferencia de ambos (este tipo de reducción se utiliza como hiperparámetro). Esta reducción es necesaria debido a que el vector resultante es pasado a una red de clasificación como se ilustra en la Figura 4.7. La red de clasificación es una red totalmente conectada con función de activación ReLU, 350 neuronas en cada capa oculta, capas totales L_{class} y una función sigmoidea final.

El modelo devuelve un valor único en el intervalo [0, 1] por cada problema de verificación de autoría, que se puede interpretar como una medida de cuánto se parecen los dos textos.

Una salida cercana a 1 dice que el modelo encuentra que ambos textos son del mismo autor. Para evaluar el modelo, se utiliza por defecto un umbral de 0.5, es decir, si la salida es mayor a 0.5 el modelo dice que ambos textos son del mismo autor, si es menor a 0.5 ambos textos son de diferentes autores y si es exactamente 0.5 se interpreta que es un problema que el modelo no puede resolver.

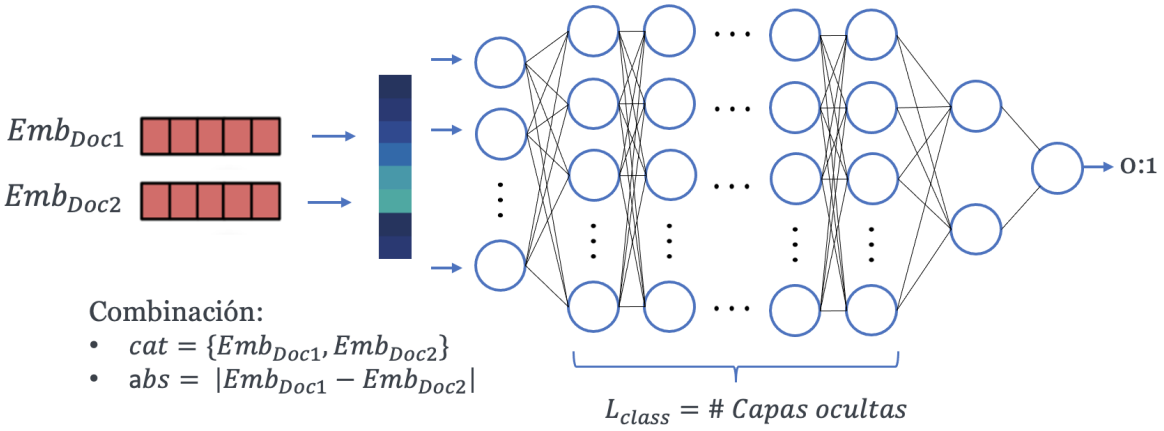

Figura 4.7: Red de clasificación.

4.3. Resumen

En este capítulo se describió la metodología propuesta para solucionar la tarea de verificación de autoría de discurso cruzado utilizando las propiedades de los grafos para representar características estilísticas de los autores en los textos. También, se describen los pasos a seguir para el pre-procesamiento de los textos antes de ser convertidos a grafos, la técnica utilizada para convertir un corpus a grafo y la importancia de las medidas utilizadas para ponderar las aristas de los grafos. Las etapas en la arquitectura de red convolucional son definidas, así como, los fundamentos de esta red convolucional basada en grafos. Finalmente, se detallan los parámetros e hiperparámetros del modelo que se utilizan en la etapa de experimentación, para ajustar la arquitectura.

Capítulo 5

Resultados y configuración experimental

En este capítulo se mostrarán y discutirán los resultados obtenidos al aplicar el método de red neuronal convolucional basada en grafos, también algunas implicaciones que se deben considerar para la experimentación. Esta tarea, supone una serie de retos, la adquisición de los datos, su procesamiento, el formato, la representación de los mismos para ser utilizados en algún modelo de aprendizaje, el análisis de los resultados, entre otros. Si bien en este trabajo no resultó un problema la adquisición de los datos, fue necesario ajustar e incrementar la información para poder utilizarla con los modelos propuestos para solucionar la tarea. En este capítulo se inicia hablando sobre la fuente en donde se obtuvieron los datos, todo lo referente al corpus utilizado para entrenar el modelo. Se describe como se realizó el pre-procesamiento de la información, así como las modificaciones realizadas. También se presentan los modelos base utilizados para comparar los resultados obtenidos con la propuesta. Finalmente se muestran los resultados que se obtienen variando los hiperparámetros del modelo propuesto y los resultados obtenidos con los modelos base.

5.1. Conjunto de datos

Para poder entrenar el modelo y evaluarlo es necesario contar con un conjunto de datos o corpus, es decir, un conjunto de textos que cumplan los requisitos de la tarea de verificación de autoría de descruzo cruzado. Dicho esto, se podría buscar en la red una serie de textos para formar el conjunto de datos y utilizarlos. Sin embargo no es tan sencillo, la construcción de un corpus no es una tarea trivial [77] como el buscar una serie de textos y unirlos en pares. Es por ello que, se decide utilizar el corpus que proporciona el PAN 2022 [101] para la tarea de verificación de autoría de discurso cruzado. Este corpus es utilizado porque cumple con los requisitos de la pregunta de investigación que se quiere resolver y cuenta con el respaldo de un comité de evaluación [28]. A continuación se explica en que consiste el conjunto de datos.

5.1.1. Corpus PAN 2022

El corpus del PAN 2022 [101] fue seleccionado ya que consta de casos de verificación de autoría de discurso cruzado. El corpus contiene textos de distintos tipos de discurso, es decir, cada uno de los textos puede ser un ensayo, un correo electrónico, mensajes de texto o memorandos comerciales. Existen un total de 1,046 textos diferentes en el corpus y fueron escritos por alrededor de 56 autores, en la Figura 5.1 se muestra un estadístico del número de textos

por autor. Todos los autores tienen edades similares entre los 18 y 22 años, y son hablantes nativos del Inglés. El tema de las muestras de texto no está restringido. Al mismo tiempo, el nivel de formalidad puede variar dentro de un determinado tipo de discurso.

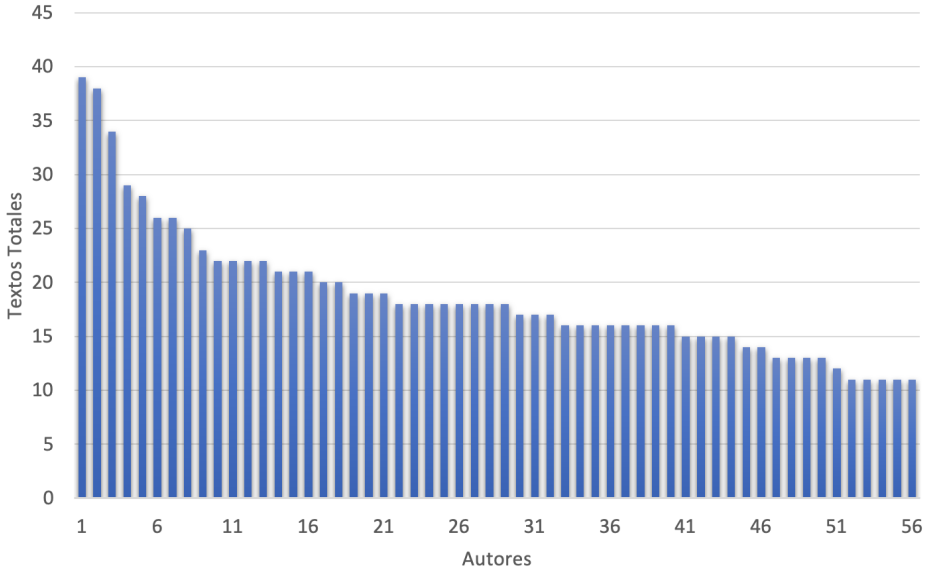

Figura 5.1: Cantidad de textos totales por autor.

La estructura del corpus consiste en pares de textos (o problemas) que pertenecen a dos tipos de discurso diferentes. Cada par tiene asignado un identificador único. Los pares están etiquetados entre los pares del mismo autor y los pares de diferentes autores. Además, se ofrece metadatos sobre el tipo de discurso para cada texto del par. Cabe destacar que hay textos que se repiten en distintos problemas o pares de textos y que existen textos en distintos problemas que son comunes en autor, es decir, el mismo autor escribió los textos. El número total de pares de texto en el conjunto de datos es de 12,264.

Dado que la longitud del texto de los mensajes y correos electrónicos pueden ser muy pequeña, cada texto perteneciente a estos tipo de discurso es en realidad una concatenación de diferentes mensajes. Se usa la etiqueta <**new**> para indicar los límites de los mensajes originales. Las nuevas líneas dentro de un texto se indican con la etiqueta <**nl**>. Además, información específica del autor y del tema (por ejemplo. entidades nombradas) ha sido sustituida con las etiquetas correspondientes.

El conjunto de datos viene con dos archivos JSON delimitados por saltos de línea codificados en UTF8. El primer archivo llamado *pairs.jsonl* y como ya se describió anteriormente, contiene pares de textos (cada par tiene una ID único) y sus etiquetas de tipo de discurso, en la Figura 5.2 se observa un ejemplo de la estructura.

```
1. {"id": "6cced668-6e51-5212-873c-717f2bc91ce6", "discourse_type": ["essay", "email"], "pair": ["Text 1...", "Text 2..."]}
2. {"id": "ae9297e9-2ae5-5e3f-a2ab-ef7c322f2647", "discourse_type": ["email", "text_message"], "pair": ["Text 3...", "Text 4..."]}
3. ...
```

Figura 5.2: Estructura del archivo *pairs.jsonl* del conjunto de datos.

El segundo archivo, *truth.jsonl*, contiene la verdad fundamental para todos los pares. La verdad básica se compone de una bandera booleana que indica si los pares de texto son del mismo autor y los identificadores del autor Figura 5.3.

```
1. {"id": "6cced668-6e51-5212-873c-717f2bc91ce6", "same": true, "authors": ["1446633", "1446633"]}
2. {"id": "ae9297e9-2ae5-5e3f-a2ab-ef7c322f2647", "same": false, "authors": ["1535385", "1998978"]}
3. ...
```

Figura 5.3: Estructura del archivo *truth.jsonl* del conjunto de datos.

5.1.2. Preprocesamiento de corpus.

Primeramente se debe realizar un limpiado del conjunto de datos eliminando textos con menos de 10 tokens (que son textos sin sentido) y también eliminando pares de textos duplicados que aparecen en los conjuntos de datos con diferentes identificaciones de problemas. En total, se eliminaron 163 problemas que estaban duplicados.

Para entrenar, validar y evaluar un modelo, es necesario dividir el corpus o conjunto de datos en tres partes. El modelo se entrena con la partición de entrenamiento, con la partición de validación se calibran los hiperparámetros del modelo, y finalmente el conjunto de prueba se utiliza para lograr una puntuación de referencia del modelo. No se utiliza ninguna muestra del conjunto de prueba para calibrar el modelo, por lo que los puntajes que se obtienen cuando se evalúa el modelo con esta partición informa sobre la capacidad de generalización del modelo.

Las particiones deben de ser ajenas en autores, esto quiere decir que los autores de los textos de cada partición no se deben de repetir en otra partición. Esto asegura que el modelo sea validado o evaluado con textos de autores que jamás ha visto el modelo y se tenga una medida confiable de la capacidad de generalización del modelo.

Dado que las parejas de textos que originalmente conforman el corpus mezclan todos los autores, es necesario hacer las particiones por autores para con esto asegurar que ningún texto de algún autor que se encuentra en una partición en particular se repita en otra. En la Figura 5.1 se muestra el total de textos por autor, entonces, tomando en cuenta esta estadística el conjunto de autores se divide en tres partes, asignando aquellos con mayor número de textos a la partición de entrenamiento y el resto repartidos para las particiones de validación y prueba. En la Tabla 5.1 se muestran la cantidad de autores y textos asignados a cada partición, esto se determinó tratando de que el conjunto de entrenamiento contara con la mayor cantidad de textos y autores posibles asignándole alrededor del 85 % de los textos.

Partición	Número de Autores	Número de Textos
Entrenamiento	47	906
Validación	4	60
Prueba	5	80

Tabla 5.1: Número de autores y textos por partición.

Al realizar las particiones por este método, muchos problemas o parejas de texto fueron eliminadas ya que utilizaban textos de autores que pertenecen a otra partición, esto produjo particiones desequilibradas, es decir, particiones con más problemas positivos que negativos, esto sesgo puede provocar que el modelo se incline más por responder de forma positiva que negativa. Para equilibrar las particiones se generaron nuevas particiones con los textos pertenecientes a cada partición considerando la restricción de que las parejas de textos de cada problema deben de ser de distinto tipo de discurso.

Para ello se aplica la siguiente metodología: Sean los conjuntos A y B los sub-conjuntos de textos de la partición agrupados por autor. Las instancias positivas y negativas se obtienen aplicando el producto cartesiano $P = A \times A$ y $N = A \times B$ respectivamente. Luego, se filtran

los pares del mismo tipo de discurso y, finalmente, se seleccionan aleatoriamente instancias positivas y negativas de los conjuntos P y N para equilibrar las particiones de entrenamiento, validación y prueba.

El nuevo conjunto de datos tiene una proporción equilibrada de problemas positivos y negativos. La Tabla 5.2 muestra el número total de problemas, el número de problemas positivos y el tamaño promedio de los texto. De cada una de las particiones.

Partición	Total de pares	Positivos	Tamaño promedio de textos
Entrenamiento	15,732	7866	560
Validación	548	274	628
Prueba	778	389	581

Tabla 5.2: Información de las particiones de entrenamiento, validación y prueba.

Con los cambios anteriores, se asegura que se tendrá el máximo número de parejas de textos para el conjunto de entrenamiento, que las particiones están balanceadas entre problemas positivos y negativos y que cada pareja pertenece a un problema de verificación de autoría de discurso cruzado.

5.2. Configuración experimental.

A continuación se presentan las configuraciones experimentales del modelo propuesto, y de una serie de modelos base. Estos modelos base son utilizados con el objetivo de comparar los resultados obtenidos aplicando el modelo de GCN. Los modelos base son propuestas basadas en métodos tradicionales y también basadas en grafos para solucionar tareas de atribución de autoría y en específico de verificación de autoría. Todos los modelos son evaluados con el mismo corpus del PAN 2022 presentado anteriormente.

5.2.1. Configuración experimental del método de verificación de autoría basado en la similitud coseno.

El método de similitud coseno, es una solución sencilla y rápida para esta tarea. En la Sección 3.2.1.1 se explica el fundamento formal de este enfoque. En este caso se representa cada uno de los documentos del corpus PAN2022 [28] utilizando el modelo de bolsa de caracteres de n-gramas, el cuál está ponderado por el esquema de pesado TF-IDF (ver Sección 2.4.1).

Utilizando la representación obtenida, la similitud coseno entre cada par de textos o problema de verificación de autoría en el conjunto de datos es calculada. Posteriormente las similitudes resultantes son optimizadas, y proyectadas a través de una simple operación de re-escalado para que funcionen como pseudo-probabilidad, indicando la probabilidad de que un par de documentos sea un par del mismo autor.

5.2.2. Configuración experimental del método de verificación de autoría basado en máquina de vectores de soporte.

El método de verificación de autoría basado en máquina de vectores de soporte (SVM por sus siglas en inglés) es un método popular en PLN y en general en el aprendizaje automático ya que es un enfoque eficaz para resolver tareas de clasificación de textos y en particular análisis

de autoría [53]. En la Sección 3.2.3.3 se explica a detalle el modelo como solución a tareas de análisis de autoría.

Para solucionar la tarea de verificación de autoría por medio de este método, cada uno de los texto del corpus PAN2022 [28] es representado como vector utilizando n-gramas (unigramas de palabras para este caso). Ya con los vectores de documentos, cada una de las parejas de texto o problemas de verificación de autoría son procesados por una etapa de reducción. La cuál consiste en obtener la diferencia de cada pareja de textos. Un clasificador SVM lineal es entrenado utilizando los vectores resultantes de la etapa de reducción.

5.2.3. Configuración experimental del método de verificación de autoría basado en compresión de textos.

Uno de los beneficios que ofrece la compresión de textos en comparación a otros métodos es que no se requiere de un preprocesamiento de los datos y es igualmente efectivo aplicado a tareas de este estilo. Este método es explicado a detalle en la Sección 3.2.2.

Para la configuración de esta solución se toman dos textos, $text_1$ y $text_2$, posteriormente se calcula la entropía cruzada del $text_2$ utilizando el modelo de compresión PPM del $text_1$ y viceversa. Luego, la media y la diferencia absoluta de las dos entropías cruzadas se utilizan para estimar una puntuación entre [0,1] que indica la probabilidad de que los dos textos estén escritos por el mismo autor. El modelo de predicción se basa en la regresión logística y se entrena utilizando la colección de casos de entrenamiento. Dado que los casos de verificación con una puntuación exactamente igual a 0.5 se consideran sin respuesta, se utiliza un radio alrededor de este valor para determinar qué rango de puntuaciones corresponderá al valor predeterminado de 0.5.

5.2.4. Configuración experimental del método de verificación de autoría basada en red siamesa basada en grafos.

Este enfoque esta basado en el propuesto en [29] para solucionar la tarea de verificación de autoría del PAN 2021 [55]. Este enfoque aprovecha las propiedades de los grafos para representar las propiedades de los textos y tratar de capturar la relación entre palabras y etiquetas POS. En esta solución, cada texto es representado como un grafo. Cada representación de texto es un grafo dirigido con tuplas de (palabra,POS) como nodos y aristas ponderadas. El cuál esta definido como el conjunto de nodos V y el conjunto de aristas E, donde cada elemento en E es una tupla (n_1,n_2,w) con n_1,$n_2 \in V$ y $w \in \mathbb{R}$ la arista ponderada. La construcción del grafo se basa en la co-ocurrencia con diferentes pesos en cada arista. Como método de aprendizaje, utilizan una red siamesa basada en grafos la cual esta compuesta por dos componentes idénticos de extracción de características que comparten pesos, una etapa de reducción vectorial y una de clasificación. Cada componente de extracción de características recibe un texto, lo transforma a grafo y regresa un vector de características del grafo; el objetivo es extraer características relevantes que puedan identificar el estilo del autor utilizando la representación gráfica del texto. En la Figura 5.4 se muestra la arquitectura de la red.

La capa de *node embedding* obtener una representación vectorial relevante de cada nodo en el grafo; cada capa está compuesta de una convolución extrema local (LEConv por sus siglas en ingés), seguida por una capa de normalización y una función de activación ReLU. Posterior a este proceso se pasa por una capa de atención global, esta capa toma la salida final del componente de extracción de características del nodo como su entrada, es decir, un grafo con un embedding en cada nodo; para obtener el vector final, hace una suma ponderada de cada vector de nodo con un coeficiente obtenido al hacer atención sobre estos mismos

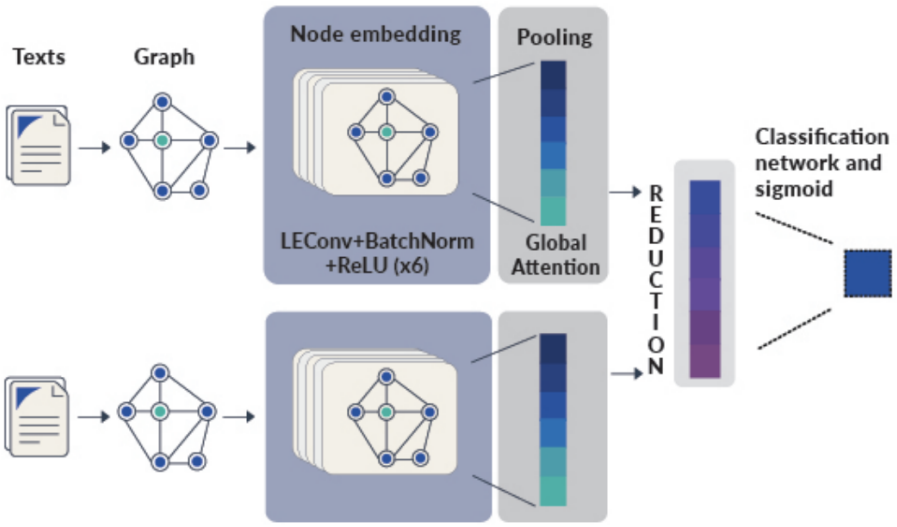

Figura 5.4: Red neuronal siamesa basada en grafos. Imagen tomada de [29].

vectores. En la etapa de reducción se obtiene el valor absoluto de la diferencia entre las salida de ambos componentes de extracción de características, este vector único es pasado a una red de clasificación. La red de clasificación es una red de cinco capas totalmente conectada con activación ReLU y función sigmoidea final. El modelo está entrenado para regresa un valor entre 0 y 1, lo cuál se interpreta como probabilidad de que ambos textos hayan sido escritos por el mismo autor o no.

5.2.5. Configuración experimental de red neuronal convolucional basada en grafos heterogéneos.

Como parte de la experimentación con la arquitectura de red neuronal convolucional basada en grafos heterogéneos, se proponen una serie de experimentos con el objetivo de estudiar el comportamiento del modelo bajo diferentes condiciones de entrenamiento y configuraciones de parámetros e hiperparámetros. La selección de hiperparámetros se realiza con base en las configuraciones que puede adoptar la representación del texto, esto con el objetivo de extraer las características estilísticas apropiadas para poder verificar la autoría de un par de textos a pesar de que estos pertenezcan a distinto tipo de discurso. Parte de esta selección de hiperparámetros esta basada en la solución que se propone en [29] la cuál también utilizan un modelo basado en grafos para solucionar la verificación de autoría. A continuación se en listan los hiperparámetros que se varían durante la experimentación.

- Tipo de grafo: En la Sección4.1 se explica como se construyen los tipos de grafo, *grafo short*, *grafo med* y *grafo full*. Cada uno de estos se utilizará para representar los textos del corpus y se ingresará en el modelo. Uno de los motivos por el cuál se definen estos tipos de grafo es para consumir menos poder de cómputo debido a que una diferencia que caracteriza a los tipos de grafo es el tamaño, por ejemplo, el *grafo short* es considerablemente mas chico al *grafo full*. Otro motivo es el uso de las etiquetas POS, como se explica en la Sección4.1 en el *grafo full* no se utilizan etiquetas POS para construir

el grafo mientras que en *grafo short* y *grafo med* si se utilizan; esta técnica es muy importante para el objetivo de verificar la autoría, porque con esto se pueden encontrar patrones de categorías gramaticales que el autor repite en distintos tipos de discurso y no simplemente de palabras.

- Palabras funcionales: Este hiperparámetro es parte del preprocesamiento de los textos. Su función es dejar o remover las palabras funcionales de los textos. Esto se asignó como hiperparámetro porque se ha demostrado [28] que las palabras funcionales son un parámetro que ayuda a verificar la autoría de un autor, es pero ello que se requiere estudiar el comportamiento del modelo cuando los textos tienen palabras funcionales o son removidas.

- Signos de puntuación: Igual que las palabras funcionales es un hiperparámetro que se aplica en el pre-procesamiento de texto. Su función es remover o no los signos de puntuación.

- Tamaño de ventana: En la Sección 4.1 se explica la construcción del grafo y se habla del tamaño de ventana el cuál es un número que define la cantidad de palabras que serán agrupadas para calcular el valor PMI (también explicado en la Sección 4.1). Un valor positivo de PMI implica una alta correlación semántica de palabras en un corpus, minetras que un valor negativo de PMI implica una poca o nula correlación semántica.

- Número de capas convolucionales: Como se menciona en la Sección 4.1 una GCN de una sola capa convolucional puede capturar información solo sobre vecinos inmediatos, cuando se apilan varias capas, se integra la información sobre vecinos más lejanos en los nodos. Es por ello que es importante considerar esta variable como un hiperparámetro.

- Combinación de embeddings: Una vez obtenidos los vectores que representan a los documentos, son procesados por una etapa de reducción en la que se combinan ambas representaciones vectoriales de los documentos, esta combinación puede ser una concatenación o deferencia absoluta de ambos vectores. Esta técnica fue utilizada en [29] la cuál proporcionó buenos resultados.

- Número de capas de clasificación: En la etapa de clasificación se varía el número de capas ocultas de una red neuronal densa.

En la Tabla 5.3 se muestran los valores para los hiperparámetros que se consideran para la experimentación. Tomando esto en cuenta se generan un total de 1,152 configuraciones para el modelo propuesto.

Tipo de grafo	Palabras funcionales	Signos de puntuación	Tamaño de ventana	Capas convolucionales	Comb.	Capas de clasificación
Short	Incluir	Incluir	10	2	cat	2
Med	No Incluir	No Incluir	20	4	abs	4
Full			30	6		6
				8		8

Tabla 5.3: Valores de hiperparámetros

5.3. Resultados experimentales.

A continuación se muestran y discuten los resultados obtenidos en entrenamiento y evaluación del modelo propuesto en la metodología. Así mismo, se presenta una comparativa con los resultados obtenidos con los modelos base propuestos. Con la finalidad de tener un modelo óptimo se realizan múltiples experimentos, variando solamente los hiperparámetros descritos en la Tabla 5.3. Por otra parte, con la finalidad de tener resultados comparables con modelos base utilizados en la verificación de autoría, se realizan experimentos con los modelos presentados en las Sub-Secciones 5.2.1, 5.2.2, 5.2.3, 5.2.4. Los modelos son evaluados con las métricas propuestas por el PAN 2022 [28]: AUC, F1 score, c@1 score, F_0.5u score, Brier score. Definidas previamente en la Sección 2.6. Por simplicidad en los resultados solo se muestra el promedio de estas medidas de evaluación.

5.3.1. Resultados con diferentes tipos de grafo.

Los resultados mostrados en las Tablas 5.4, 5.5 y 5.6 son pruebas que se realizan con el objetivo de observar el comportamiento del modelo al variar las capas tanto de convolución como de clasificación para cada uno de los tipos de grafo (Cada tabla corresponde a los resultados obtenidos con un tipo de grafo). Dicho lo anterior, en la Tabla 5.4 se muestran los resultados del modelo utilizando el *grafo short*. En este conjunto de pruebas se mantuvieron fijos los hiperparámetros restantes, los cuales son explicados a continuación.

Para generar el grafo se utiliza un tamaño de ventana de 30 tokens, se preprocesa el texto removiendo signos de puntuación y se conservan las palabras funcionales. En la etapa de reducción vectorial se aplica concatenación de vectores, para combinar los embeddings que representan a cada uno de los textos a un solo embedding.

Analizando la Tabla 5.4 se observa que en estos experimentos los resultados mas altos obtenidos son al utilizar **2** capas convolucionales en la etapa de extracción de características. A su vez se observa que conforme las capas convolucionales van aumentando los resultados con el conjunto de prueba van disminuyendo. En el caso de la variación de capas de clasificación el resultado mas alto obtenido es con **8** capas, aquí se observa un comportamiento contrario, conforme aumentan las capas los resultados tienen una leve mejoría.

En la Tabla 5.5 se muestran los resultados de los experimentos pero en estos se utiliza el *grafo Med* para representar los textos del corpus. En este caso los resultados mas altos obtenidos son al utilizar **4** capas convolucionales en la etapa de extracción de características. Se observa un comportamiento similar que con el grafo Short, conforme las capas convolucionales van aumentando los resultados con el conjunto de prueba van disminuyendo. En el caso de la variación de capas de clasificación el resultado mas alto obtenido es con **8** capas, y conforme aumentan las capas los resultados tienen una leve mejoría.

Por último en la Tabla 5.6 se muestran los resultados de los experimentos utilizando el *grafo Full* como representación de los textos del corpus. Los resultados mas altos obtenidos con esta configuración son al utilizar **8** capas convolucionales en la etapa de extracción de características y **8** capas de clasificación. El comportamiento del modelo cambia un poco con respecto a los anteriores experimentos utilizando el *grafo Short* y *grafo Med*. Se observan mejorías cuando las capas convolucionales aumentan al igual que las capas de clasificación, sin embargo, no se llegan a resultados similares a los obtenidos con los otros tipos de grafos.

Los experimentos anteriores se realizaron para analizar el comportamiento del modelo variando hiperparámetros de las etapas de clasificación y extracción de características. Las

Capas convolucionales	Capas de clasificación	Resultado validación	Resultado en prueba
2	2	0.7024	0.7905
	4	0.6908	0.7943
	6	0.6989	0.8001
	8	0.7013	**0.8029**
4	2	0.6948	0.7919
	4	0.7121	0.7982
	6	0.6954	0.7958
	8	0.7082	0.7989
6	2	0.6965	0.7811
	4	0.7047	0.785
	6	0.6905	0.7847
	8	0.7067	0.7834
8	2	0.7047	0.761
	4	0.7282	0.7709
	6	0.7074	0.7675
	8	0.7186	0.7718

Tabla 5.4: Resultados para los diferentes números de capas en clasificación y en convolución utilizando el grafo Short.

Capas convolucionales	Capas de clasificación	Resultado validación	Resultado en prueba
2	2	0.7026	0.794
	4	0.6894	0.7912
	6	0.7217	0.7752
	8	0.7182	0.8092
4	2	0.6726	0.7922
	4	0.6907	0.7854
	6	0.7231	0.7874
	8	0.7141	**0.8104**
6	2	0.6994	0.7901
	4	0.7142	0.7988
	6	0.7216	0.8025
	8	0.7086	0.7997
8	2	0.7147	0.7809
	4	0.7297	0.7846
	6	0.7173	0.7936
	8	0.728	0.7784

Tabla 5.5: Resultados para los diferentes números de capas en clasificación y en convolución utilizando el grafo Med.

siguientes pruebas están enfocadas en observar el comportamiento del modelo al variar hiperparámetros que modifican el preprocesamiento del texto, la construcción del grafo y la

Capas convolucionales	Capas de clasificación	Resultado validación	Resultado en prueba
2	2	0.621	0.6659
	4	0.6341	0.6533
	6	0.6344	0.6776
	8	0.5743	0.7066
4	2	0.6653	0.6275
	4	0.51	0.6774
	6	0.5824	0.6222
	8	0.566	0.6396
6	2	0.6648	0.7209
	4	0.561	0.7327
	6	0.5723	0.6818
	8	0.6501	0.6936
8	2	0.6537	0.719
	4	0.6245	0.739
	6	0.63	0.7182
	8	0.6354	**0.743**

Tabla 5.6: Resultados para los diferentes números de capas en clasificación y en convolución utilizando el grafo Full.

combinación de embeddings.

Dado que los siguientes experimentos están enfocados en analizar el comportamiento del modelo variando hiperparámetros que modifican de alguna forma la construcción del grafo. Las capas de convolución se dejan fijas en 4 capas y 8 para las capas de clasificación.

En las Tabla 5.7 se muestran los resultados utilizando como entrada el *grafo Short* y aplicando el método de concatenación como combinación de embeddings de documento. En las dos primeras filas de la Tabla 5.7 se observan los resultados cuando no se remueven las palabras funcionales de los textos, mientras que en las dos últimas filas si son removidas las palabras funcionales, esto es importante ya que las palabras funcionales pueden ayudar en la tarea de verificación de autoría.

En los experimentos de la Tabla 5.7 cuando se incluyen palabras funcionales, el mejor resultado se obtiene cuando se remueven los signos de puntuación de los textos, se utiliza un tamaño de ventana de **30** para construir el grafo y como método de combinación de embeddings se aplica la concatenación.

En los experimentos de la Tabla 5.7 cuando las palabras funcionales son removidas el mejor resultado a diferencia de los resultados anteriores se obtiene cuando se conservan los signos de puntuación de los textos, se utiliza un tamaño de ventana para construir el grafo de **30** y como método de combinación de embeddings se aplica la concatenación.

En la siguiente Tabla 5.8, se realizan los experimentos la diferencia es que en este caso se cambia el método de combinación de embeddings de documento. En el caso anterior se aplicó la concatenación y para estos experimentos se utiliza la diferencia absoluta. En estos resultados el mayor obtenido es cuando se tiene una ventana de **30** y se incluyen signos de puntuación, aunque claramente se observa que los resultados obtenidos con el método de concatenación son superiores. Con este conjunto de resultados se concluyen las pruebas realizadas con el *grafo Short*. A continuación se mostrarán los resultados de los mismos experimentos pero utilizando el *grafo Med*.

PF	SP	Ventana	Validación	Prueba
Si	No	10	0.7057	0.7953
		20	0.702	0.7953
		30	0.7082	**0.7989**
	Si	10	0.706	0.7942
		20	0.7044	0.7873
		30	0.6961	0.7881
No	No	10	0.7083	0.7917
		20	0.705	0.7991
		30	0.7276	0.7929
	Si	10	0.7196	0.794
		20	0.7146	0.7945
		30	0.7158	**0.7959**

Tabla 5.7: Resultados variando los valores de tamaño de ventana, signos de puntuación (SP) del *grafo Short* e incluyendo/removiendo palabras funcionales (PF). Utilizando la concatenación como método de combinación de embeddings.

PF	SP	Ventana	Validación	Prueba
Si	No	10	0.5786	0.6108
		20	0.5754	0.6125
		30	0.5636	0.6174
	Si	10	0.5926	0.6203
		20	0.584	0.6079
		30	0.5396	**0.6246**
No	No	10	0.5863	0.6119
		20	0.5878	0.6114
		30	0.5772	**0.6178**
	Si	10	0.5682	0.6139
		20	0.596	0.609
		30	0.5767	0.6128

Tabla 5.8: Resultados variando los valores de tamaño de ventana, signos de puntuación (SP) del *grafo Short* e incluyendo/removiendo palabras funcionales (PF). Utilizando la diferencia absoluta como método de combinación de embeddings.

En la Tabla 5.9 se muestran los resultados utilizando como entrada el *grafo Med* y aplicando el método de concatenación como combinación de embeddings de documento. En las dos primeras filas de la Tabla 5.9 se observan los resultados cuando no se remueven las palabras funcionales de los textos, mientras que en las dos últimas filas si son removidas las palabras funcionales. Comparando los resultados con los obtenidos con el *grafo Short*, claramente son mas altos. El resultado mas alto obtenido es cuando se remueven los signos de puntuación, se utiliza un tamaño de ventana de 30 tokens, incluyendo palabras funcionales y utilizando la concatenación como método de combinación de embeddings.

Aunque los resultados de las útimas dos filas en de Tabla 5.9 no son superiores a los experimentos cuando se incluyen palabras funcionales (primeras dos filas), siguen estando por encima de los obtenidos con el *grafo Short*. Esto indica que el *grafo Med* brinda mayor información de los estilos de escritura en los diferentes tipos de discurso.

PF	**SP**	**Ventana**	**Validación**	**Prueba**
Si	No	10	0.7133	0.805
		20	0.7013	0.8048
		30	0.7141	**0.8104**
	Si	10	0.7036	0.7725
		20	0.6961	0.7675
		30	0.7089	0.7846
No	No	10	0.7016	0.7916
		20	0.7001	0.7998
		30	0.71	**0.7981**
	Si	10	0.6821	0.769
		20	0.6933	0.7967
		30	0.7032	0.7939

Tabla 5.9: Resultados variando los valores de tamaño de ventana, signos de puntuación (SP) del *grafo med* e incluyendo/removiendo palabras funcionales (PF). Utilizando la concatenación como método de combinación de embeddings.

En la siguiente Tabla 5.10 se realizan los experimentos con la diferencia que en este caso el método de combinación de embeddings es la diferencia absoluta. Este método de combinación no resulta favorable, los resultados disminuyen considerablemente en todos los casos.

Ahora en la Tabla 5.11 se muestran los resultados utilizando como entrada el *grafo Full* y aplicando el método de concatenación como combinación de embeddings de documento. En las dos primeras filas de la Tabla 5.11 se observan los resultados cuando no se remueven las palabras funcionales de los textos, mientras que en las dos últimas filas si son removidas las palabras funcionales. Comparando los resultados con los obtenidos con el *grafo Med*, estos disminuyen. El resultado mas alto obtenido es cuando se remueven los signos de puntuación, se utiliza un tamaño de ventana de 10 tokens, se remueven palabras funcionales y utilizando la concatenación como método de combinación de embeddings.

Por último en las Tabla 5.12 se presentan los resultados de los experimentos aplicando como método de combinación de embeddings la diferencia absoluta. En este caso del *grafo Full* no varían tanto los resultados con respecto al método de concatenación, como lo fue en el caso del *grafo Med* y *grafo Short*. Los resultados son similares a los obtenidos en la Tabla 5.11, solo disminuyen ligeramente, esto nos indica que para este tipo de tarea el método de concatenación

PF	SP	Ventana	Validación	Prueba
Si	No	10	0.5941	0.6093
		20	0.5685	0.6097
		30	0.566	**0.6262**
	Si	10	0.5755	0.6211
		20	0.5436	0.618
		30	0.5748	0.6117
No	No	10	0.6031	0.5967
		20	0.6023	0.6043
		30	0.6056	0.6009
	Si	10	0.5739	**0.6187**
		20	0.5794	0.6103
		30	0.6095	0.6139

Tabla 5.10: Resultados variando los valores de tamaño de ventana, signos de puntuación (SP) del *grafo med* e incluyendo/removiendo palabras funcionales (PF). Utilizando la diferencia absoluta como método de combinación de embeddings.

PF	SP	Ventana	Validación	Prueba
Si	No	10	0.5919	0.6196
		20	0.5503	0.6289
		30	0.566	0.6396
	Si	10	0.5957	0.6442
		20	0.6458	0.6197
		30	0.5942	**0.6543**
No	No	10	0.5933	**0.673**
		20	0.5721	0.664
		30	0.5621	0.668
	Si	10	0.6381	0.6548
		20	0.5857	0.6376
		30	0.6343	0.6471

Tabla 5.11: Resultados variando los valores de tamaño de ventana, signos de puntuación (SP) del *grafo full* e incluyendo/removiendo palabras funcionales (PF). Utilizando la concatenación como método de combinación de embeddings.

de embeddings resulta mas favorable que la diferencia absoluta.

PF	SP	Ventana	Validación	Prueba
Si	No	10	0.511	**0.6204**
		20	0.5659	0.6098
		30	0.6026	0.6021
	Si	10	0.5766	0.5974
		20	0.5877	0.6023
		30	0.5945	0.6022
No	No	10	0.5546	0.6037
		20	0.5824	0.6005
		30	0.5895	0.5996
	Si	10	0.592	**0.6071**
		20	0.5997	0.5953
		30	0.5882	0.5976

Tabla 5.12: Resultados variando los valores de tamaño de ventana, signos de puntuación (SP) del *grafo full* e incluyendo/removiendo palabras funcionales (PF). Utilizando la diferencia absoluta como método de combinación de embeddings.

5.3.2. Resultados variando tamaño de embeddings de documento.

Los experimentos mostrados en la Tabla 5.13 se realizan con la finalidad de observar el comportamiento del modelo cuando se varía el tamaño de los embeddings de documento, es decir, la representación de los textos en cada uno de los nodos del grafo. Para mayor información de como se generan las representaciones de los textos y los grafos consultar la Sección 4.1.

Para realizar estos experimentos se utiliza la siguiente configuración de hiperparámetros, se aplica concatenación como método de combinación de embeddings, se remueven signos de puntuación, se incluyen palabras funcionales, se utiliza un tamaño de ventana de 30 tokens, se aplica el tipo de grafo Med, 4 copas de convolución y 8 en clasificación. Esta configuración es utilizada, ya que fue la que mostró mejor desempeño cuando se experimentó con los tipos de grafos y los distintos valores de hiperparámetros.

Como se muestra en la Tabla 5.13 y en la Figura 5.5 el resultado mas alto obtenido es cuando se tiene un tamaño de embedding de 100. Cuando se aumenta el tamaño los resultados del modelo comienzan a decrecer. Los tamaños de embeddings demasiado bajos pueden no propagar bien la información de la etiqueta a todo el gráfico, mientras que los tamaños mas altos no mejoran el rendimiento de la clasificación y pueden costar más tiempo de entrenamiento [115].

5.3.3. Resultados finales.

En la etapa de evaluación la configuración que mejor desempeño demostró fue al utilizar un *grafo Med*, un tamaño de ventana de 30, en la etapa de extracción de características 4 capas convolucionales, 8 capas en la etapa de clasificación, incluyendo palabras funcionales, removiendo signos de puntuación y utilizando la concatenación como método de combinación de embeddings. El promedio de las métricas de evaluación anteriormente explicadas en prueba fue de 0.8104.

Tamaño de embeddings	Prueba
50	0.725
60	0.7983
100	**0.8062**
150	0.8004
200	0.7642
250	0.7519
300	0.7749

Tabla 5.13: Resultados variando el tamaño de embeddings. Utilizando concatenación como combinación de embeddings, se remueven signos de puntuación, se incluyen palabras funcionales, un tamaño de ventana de 30 tokens, tipo de *grafo Med*, 4 capas de convolución y 8 en clasificación.

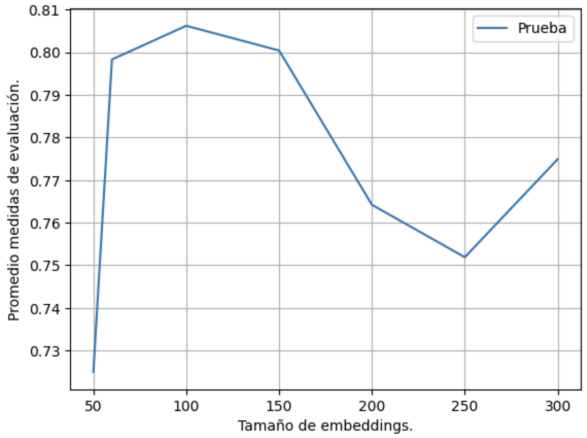

Figura 5.5: Comportamiento de resultados variando el tamaño de embeddings.

En la figura 5.6 se muestra el comportamiento de la pérdida en validación y entrenamiento. Se puede observar que el modelo se sobre ajusta pasadas las 20 épocas de entrenamiento.

Por último en la Tabla 5.14 se muestra los mejores resultados obtenidos con los métodos tradicionales y el modelo propuesto anteriormente explicados en la Sección 5.2. En la primer columna de la tabla se listan los métodos utilizados para resolver el problema de verificación de autoría y en la segunda columna el mejor resultado del promedio de las medidas de evaluación (AUC, F1, c@1, F_0.5u, Brier), obtenido con cada uno de los modelos en el conjunto de prueba.

- Red siamesa basada en grafos 5.2.4: Este modelo utiliza una estructura de red siamesa que se basa en la representación basada en grafos de co-ocurrencia de los textos. Obtuvo un puntaje promedio de evaluación de 0.600.

- Similitud coseno 5.2.1: Este modelo calcula la similitud coseno entre los vectores de características de los textos para verificar la autoría. Obtuvo un puntaje promedio de evaluación de 0.570.

- Máquina de vectores de soporte 5.2.2: Este modelo utiliza el algoritmo de Máquina de

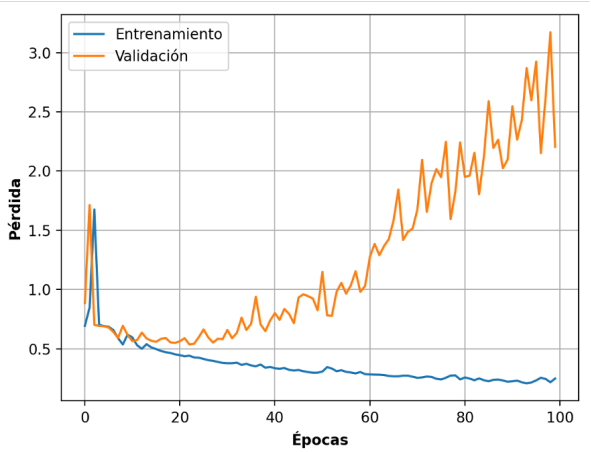

Figura 5.6: Comportamiento de la pérdida en entrenamiento y validación.

Vectores de Soporte (SVM) para clasificar los textos en términos de escritura del autor. Obtuvo un puntaje promedio de evaluación de 0.603.

- Compresión de textos 5.2.3: Este modelo se basa en la idea de que los estilos de escritura de los autores se pueden identificar mediante la compresión de los textos. Obtuvo un puntaje promedio de evaluación de 0.552.
- GCN 5.2.5: Este modelo utiliza una Red Neuronal Convolucional Basada en Grafos (GCN) para analizar y clasificar los textos en términos del estilo de escritura del autor. Obtuvo el puntaje promedio más alto de evaluación, con 0.810.

Método	Resultado en prueba
Red siamesa basada en grafos	0.600
Similitud coseno	0.570
Máquina de vectores de soporte	0.603
Compresión de textos	0.552
GCN	**0.810**

Tabla 5.14: Mejores resultados obtenidos con modelos tradicionales y con la GCN para la verificación de autoría de discurso cruzado.

En la Tabla 5.14 se puede observar que el modelo GCN fue el más efectivo en la tarea de verificación de autoría, con un puntaje promedio de evaluación de 0.810. Sin embargo, la red siamesa basada en grafos también obtuvo un resultado aceptable de 0.600. Esto sugiere que la utilización de redes neuronales convolucionales basadas en grafos pueden ser muy eficientes para abordar este problema. Sin embargo, el modelo de similitud coseno y la compresión de textos mostraron un rendimiento relativamente más bajo, con puntajes promedio de evaluación de 0.570 y 0.552 respectivamente. Esto sugiere que estos enfoques pueden no ser tan efectivos para abordar el problema de verificación de autoría en comparación con los otros modelos mencionados.

En general, se puede concluir que los modelos basados en redes neuronales convolucionales y en estructuras de red siamesa basada en grafos parecen ser las técnicas más prometedoras

para la verificación de autoría en este contexto específico. Sin embargo, es importante tener en cuenta que estos resultados son promedios y pueden variar dependiendo de los datos utilizados y las configuraciones específicas de cada modelo.

5.4. Resumen

A partir de los resultados presentados en éste capítulo podemos resumir que nuestra metodología tiene ventaja con respecto a los modelos tradicionales. Los enfoques basados en grafos han sido utilizados para representar textos y capturar el estilo de escritura de los autores, así como, características sintácticas las cuales no se pueden representar con los métodos tradicionales de representación de textos. En esta sección se presentan características del conjunto de datos utilizado para evaluar el modelo así como algunas modificaciones realizadas a este, se muestran una serie de experimentos que se realizaron utilizando el modelo propuesto y como se comporta cuando se varían distintos hiperparámetros. Las pruebas fueron organizadas de modo que se varían los hiperparámetros que modifican el preprocesameinto de los textos, la forma de construir el grafo, las capas de la red de convolución y la red densa de clasificación y el método de combinación de embeddings para formar las parejas que representan cada uno de los problemas de verificación de autoría utilizando los embeddings de documento. También se definen y evalúan algunos métodos tradicionales utilizados para comparar los resultados obtenidos con el modelo propuesto.

Capítulo 6

Conclusiones y trabajo futuro

En este trabajo de tesis se expone un enfoque general para resolver la tarea de verificación de autoría de discurso cruzado basado en el estilo de escritura del autor. Este enfoque está basado en el uso de grafos para representar los textos las características que definen el estilo de escritura de los autores en los distintos tipos de discurso.

Se realizaron una serie de modelos base los cuales utilizan distintos métodos para solucionar la tarea de verificación de autoría como, vectores de soporte, bolsa de palabras, compresión de textos y basados en grafos. Los resultados obtenidos en los experimentos realizados con los modelos base tienen un desempeño bajo en general cuando se utilizan para la variante de discurso cruzado. Sin embargo estos mismos modelos base muestran un buen desempeño en la tarea de verificación de autoría pero cuando el tipo de discurso es el mismo.

El modelo propuesto esta basado en una arquitectura basada en grafos compuesta por una etapa de extracción de características en la que se utiliza un red neuronal convolucional para grafos y una etapa de clasificación en la que se utiliza una red densa. El modelo es entrenado con un conjunto de datos obtenido del PAN2022 [28]. Para evaluar el modelo se variaron distintos hiperparámetros que modifican el preprocesamiento de los textos, la construcción del grafo, el método de combinación de embeddings de documento y las capas de convolución y de clasificación. Se observó que el tipo de grafo utilizado juega un papel muy importante ya que el tipo de grafo se basa en el número de categorías gramaticales utilizadas para representar el texto en lugar de las palabras. Cuando no se utilizaban categorías gramaticales para representar el texto y construir el grafo el rendimiento del modelo decrece al igual que cuando todas las palabras son reemplazadas por su etiqueta POS. El mejor valor obtenido es cuando se definen un conjunto de etiquetas POS a utilizar como el definido en [30]. La idea detrás de esta técnica es reducir la información léxica del texto y enfocarse en las estructuras gramaticales y sintácticas más generales. Al igual que al convertir el texto en un grafo, se pueden analizar las relaciones entre las palabras, y las categorías gramaticales, lo que puede proporcionar cierta información sobre la autoría del texto. Las categorías gramaticales que se utilizan para reemplazar las palabras son las siguientes:

- Adjetivos: Estos son modificadores que describen características o cualidades de los sustantivos.
- Sustantivos: Estas son palabras que se utilizan para identificar personas, lugares, cosas o ideas.
- Adverbios: Estos son modificadores que proporcionan información adicional sobre los verbos, adjetivos u otros adverbios en términos de tiempo, lugar, manera, etc.

- Verbos: Estas son palabras que expresan acciones, eventos o estados.
- Números cardinales: Estas son palabras que se utilizan para contar o representar una cantidad numérica.
- Palabras extranjeras: Estas son palabras tomadas directamente de otros idiomas.
- Marcador de elementos de lista: Estos son marcadores utilizados para indicar elementos de una lista.
- Símbolos: Estos son caracteres o signos que tienen un significado específico. Por ejemplo, en lugar de utilizar el símbolo "&.[en] un texto, se usaría la categoría gramatical "SYM".

Las palabras que se reemplazan por su categoría gramatical en este enfoque no retienen sus características léxicas específicas, como el significado o la forma exacta de la palabra original. En cambio, se sustituyen por la categoría gramatical correspondiente, lo que permite analizar las estructuras y relaciones gramaticales dentro del texto de una manera más abstracta. En lugar de preservar las características individuales de cada palabra, se enfoca en la función gramatical que desempeñan dentro de la oración. Por ejemplo, si se reemplaza la palabra rápidamente"por la categoría gramatical .[A]dverbios", se pierde la información sobre la velocidad o la manera específica en que ocurrió una acción. El objetivo de esta técnica es capturar patrones gramaticales más generales y analizar la estructura sintáctica del texto, en lugar de centrarse en los detalles semánticos o léxicos. Al transformar el texto en un grafo utilizando estas categorías gramaticales, se pueden identificar las conexiones y relaciones entre las palabras según su función gramatical en la oración.

Otro parámetro a considerar es el número de capas convolucionales utilizadas en la etapa de extracción de características, se observó que conforme aumentan el número de capas el rendimiento del modelo decrece al igual que en la solución propuesta en [115], con nuestros experimentos se encontró que con 4 capas el modelo tiene un buen rendimiento si las capas aumentan el rendimiento no mejora. El uso de palabras funcionales es otro parámetro del modelo. Las palabras funcionales son aquellas que desempeñan un papel gramatical o estructural en una oración, pero a menudo tienen un significado léxico limitado. Cuando se eliminan las palabras funcionales en el proceso de construcción del grafo, se pueden producir cambios significativos en la estructura y las relaciones del grafo resultante. Estas palabras funcionales suelen ser de uso frecuente y aparecen en muchas oraciones sin aportar un significado léxico sustancial. Al eliminar estas palabras del grafo, se puede obtener una representación más simplificada y centrada en las palabras con mayor carga léxica, como sustantivos, verbos y adjetivos. Esto puede ayudar a enfocarse en las relaciones semánticas y sintácticas más importantes dentro del texto. En los experimentos realizados cuando las palabras funcionales eran removidas de los textos los resultados decrecían, esto se puede deber a que, los autores suelen utilizar ciertas palabras funcionales con mayor frecuencia que otros, entonces esto puede ser una característica estilística que defina el estilo de un autor.

Con respecto a los embeddings de documento en los experimentos se observó que es importante definir un método de combinación el cuál beneficie al modelo a no perder información de ambos textos, ya que para los experimentos se utilizó tanto la concatenación como la diferencia absoluta y con el segundo método el rendimiento del modelo decrecía significativamente.

Todos los modelos fueron entrenados y evaluados con particiones del conjunto de datos del PAN2022 [28], es importante mencionar que estas particiones fueron generadas de tal manera que fueran ajenas en autores, es decir, no compartieran textos de los mismos autores.

El modelo propuesto basado en grafo heterogéneo, arroja mejores resultados en comparación a los modelo base tradicionales. Verificar dos textos de distinto tipo de discurso, requiere de una abstracción profunda de la evidencia que se tiene. Los modelos basados en bolsa de palabras o n-gramas se ven limitados, ya que el conteo de frecuencia de término a nivel carácter o palabras no es suficiente debido a que no captura las características sintácticas del texto. Por otra parte, el modelo propuesto con la ayuda de los grafos es posible capturar este tipo de características estilísticas.

Se concluye que se encuentra favorable la aplicación del modelo propuesto para la tarea de verificación de autoría de discurso cruzado. Ya que el el grafo heterogéneo logra modelar el estilo de escritura del autor a través de distintos tipos de discurso. El enfoque propuesto tiene un mejor rendimiento cuando se definen un conjunto de etiquetas POS apropiadas para definir el grafo y con esto capturar solo la información necesaria de los textos que definan el estilo del autor.

6.1. Trabajo futuro

El modelo fue entrenado, validado y probado con un solo conjunto de datos proporcionado por el PAN2022 [28], aunque se realizaron modificaciones para tener particiones con autores ajenos. Uno de los puntos que se pueden abordar en trabajos futuros, es utilizar dos distintos conjuntos de datos, uno para entrenar y validar el modelo y otro para probar el modelo e incluso que ambos conjuntos de textos cuenten con distintos tipos de discurso. En este trabajo se utilizaron textos de longitud promedio de 560 y máximo 628 caracteres. Para trabajos futuros habría que evaluar si la metodología presenta resultados equiparables cuando se usan documentos de longitud mayor que 628.

Bibliografía

[1] O. Abdel-Hamid, A.-r. Mohamed, H. Jiang, L. Deng, G. Penn, and D. Yu. Convolutional neural networks for speech recognition. *IEEE/ACM Transactions on audio, speech, and language processing*, 22(10):1533–1545, 2014.

[2] A. Antonia, H. Craig, and J. Elliott. Language chunking, data sparseness, and the value of a long marker list: explorations with word n-grams and authorial attribution. *Literary and Linguistic Computing*, 29(2):147–163, 2014.

[3] S. Argamon, M. Koppel, J. Fine, and A. R. Shimoni. Gender, genre, and writing style in formal written texts. *Text-The Hague Then Amsterdam Then Berlin-*, 23(3):321–346, 2003.

[4] G. Barlas and E. Stamatatos. Cross-domain authorship attribution using pre-trained language models. In *Artificial Intelligence Applications and Innovations: 16th IFIP WG 12.5 International Conference, AIAI 2020, Neos Marmaras, Greece, June 5–7, 2020, Proceedings, Part I 16*, pages 255–266. Springer, 2020.

[5] J. Bastings, I. Titov, W. Aziz, D. Marcheggiani, and K. Sima'an. Graph convolutional encoders for syntax-aware neural machine translation. *arXiv preprint arXiv:1704.04675*, 2017.

[6] P. W. Battaglia, J. B. Hamrick, V. Bapst, A. Sanchez-Gonzalez, V. Zambaldi, M. Malinowski, A. Tacchetti, D. Raposo, A. Santoro, R. Faulkner, et al. Relational inductive biases, deep learning, and graph networks. *arXiv preprint arXiv:1806.01261*, 2018.

[7] J. Bevendorff, B. Stein, M. Hagen, and M. Potthast. Generalizing unmasking for short texts. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 654–659, 2019.

[8] C. M. Bishop and N. M. Nasrabadi. *Pattern recognition and machine learning*, volume 4. Springer, 2006.

[9] D. C. Blair et al. Information retrieval, cj van rijsbergen. london: Butterworths; 1979: 208 pp. *Journal of the American Society for Information Science*, 30(6):374–375, 1979.

[10] V. Bobicev. Authorship detection with ppm. In *CLEF2013 Working Notes*. CEUR, 2013.

[11] B. Boenninghoff, R. M. Nickel, S. Zeiler, and D. Kolossa. Similarity learning for authorship verification in social media. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2457–2461. IEEE, 2019.

[12] G. W. Brier et al. Verification of forecasts expressed in terms of probability. *Monthly weather review*, 78(1):1–3, 1950.

[13] P. F. Brown, S. A. Della Pietra, V. J. Della Pietra, J. C. Lai, and R. L. Mercer. An estimate of an upper bound for the entropy of english. *Computational Linguistics*, 18(1):31–40, 1992.

[14] J. Brownlee. A gentle introduction to cross-entropy for machine learning. *Machine Learning Mastery. https://machinelearningmastery. com/cross-entropy-for-machine-learning*, 2019.

[15] J. Bruna, W. Zaremba, A. Szlam, and Y. LeCun. Spectral networks and locally connected networks on graphs. *arXiv preprint arXiv:1312.6203*, 2013.

[16] J. Burrows. 'delta': a measure of stylistic difference and a guide to likely authorship. *Literary and linguistic computing*, 17(3):267–287, 2002.

[17] H. Cai, V. W. Zheng, and K. Chang. A comprehensive survey of graph embedding: Problems, techniques and applications. corr abs/1709.07604 (2017). *arXiv preprint arxiv:1709.07604*, 2017.

[18] Y. Cai and X. Cheng. Biomedical named entity recognition with tri-training learning. In *2009 2nd International Conference on Biomedical Engineering and Informatics*, pages 1–5. IEEE, 2009.

[19] E. Castillo, O. Cervantes, D. Vilari, and B. David. Author verification using a graph-based representation. *International Journal of Computer Applications*, 123(14), 2015.

[20] E. Castillo, O. Cervantes, and D. Vilarino. Text analysis using different graph-based representations. *Computación y Sistemas*, 21(4):581–599, 2017.

[21] C. E. Chaski. Who's at the keyboard? authorship attribution in digital evidence investigations. *International journal of digital evidence*, 4(1):1–13, 2005.

[22] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

[23] J. Cleary and I. Witten. Data compression using adaptive coding and partial string matching. *IEEE transactions on Communications*, 32(4):396–402, 1984.

[24] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa. Natural language processing (almost) from scratch. *Journal of machine learning research*, 12(ARTICLE):2493–2537, 2011.

[25] M. Defferrard, X. Bresson, and P. Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. *Advances in neural information processing systems*, 29, 2016.

[26] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[27] S. H. Ding, B. C. Fung, F. Iqbal, and W. K. Cheung. Learning stylometric representations for authorship analysis. *IEEE transactions on cybernetics*, 49(1):107–121, 2017.

[28] Efstathios Stamatatos and Mike Kestemont and Krzysztof Kredens and Piotr Pezik and Annina Heini and Janek Bevendorff and Martin Potthast and Benno Stein. Overview of the Authorship Verification Task at PAN 2022. In *CLEF 2022 Labs and Workshops, Notebook Papers*, CEUR Workshop Proceedings, 2022.

[29] D. Embarcadero-Ruiz, H. Gómez-Adorno, A. Embarcadero-Ruiz, and G. Sierra. Graph-based siamese network for authorship verification. *Mathematics*, 10(2):277, 2022.

[30] D. Embarcadero-Ruiz, H. Gómez-Adorno, A. Embarcadero-Ruiz, and G. Sierra. Graph-based siamese network for authorship verification. *Mathematics*, 10(2):277, 2022.

[31] S. Evert, T. Proisl, F. Jannidis, I. Reger, S. Pielström, C. Schöch, and T. Vitt. Understanding and explaining delta measures for authorship attribution. *Digital Scholarship in the Humanities*, 32(suppl_2):ii4–ii16, 2017.

[32] T. Fawcett. An introduction to roc analysis. *Pattern recognition letters*, 27(8):861–874, 2006.

[33] G. Frantzeskou, E. Stamatatos, S. Gritzalis, and S. Katsikas. Effective identification of source code authors using byte-level information. In *Proceedings of the 28th international conference on Software engineering*, pages 893–896, 2006.

[34] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. corr, abs/1311.2524. *arXiv preprint arXiv:1311.2524*, 2013.

[35] H. Gómez-Adorno, G. Sidorov, D. Pinto, D. Vilariño, and A. Gelbukh. Automatic authorship detection using textual patterns extracted from integrated syntactic graphs. *Sensors*, 16(9):1374, 2016.

[36] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[37] I. J. Goodfellow, D. Erhan, P. L. Carrier, A. Courville, M. Mirza, B. Hamner, W. Cukierski, Y. Tang, D. Thaler, D.-H. Lee, et al. Challenges in representation learning: A report on three machine learning contests. In *Neural Information Processing: 20th International Conference, ICONIP 2013, Daegu, Korea, November 3-7, 2013. Proceedings, Part III 20*, pages 117–124. Springer, 2013.

[38] O. Halvani, L. Graner, and R. Regev. Taveer: an interpretable topic-agnostic authorship verification method. In *Proceedings of the 15th International Conference on Availability, Reliability and Security*, pages 1–10, 2020.

[39] O. Halvani, M. Steinebach, and R. Zimmermann. Authorship verification via k-nearest neighbor estimation. *Notebook PAN at CLEF*, 2013.

[40] S. H. Harvey. Author verification using ppm with parts of speech tagging. In *CLEF (Working Notes)*, pages 1063–1068, 2014.

[41] M. Henaff, J. Bruna, and Y. LeCun. Deep convolutional networks on graph-structured data. *arXiv preprint arXiv:1506.05163*, 2015.

[42] G. E. Hinton, A. Krizhevsky, and I. Sutskever. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25(1106-1114):1, 2012.

[43] S. Hochreiter, Y. Bengio, P. Frasconi, J. Schmidhuber, et al. Gradient flow in recurrent nets: the difficulty of learning long-term dependencies, 2001.

[44] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[45] D. I. Holmes. The evolution of stylometry in humanities scholarship. *Literary and linguistic computing*, 13(3):111–117, 1998.

[46] M. Hosseinia and A. Mukherjee. A parallel hierarchical attention network for style change detection. In *CLEF 2018-Conference and Labs of the Evaluation Forum*, volume 2125, 2018.

[47] F. Iqbal, M. Debbabi, and B. C. Fung. *Machine learning for authorship attribution and cyber forensics*. Springer, 2020.

[48] S. Jacques. Machine learning methods for stylometry, 2020.

[49] F. Jafariakinabad, S. Tarnpradab, and K. A. Hua. Syntactic recurrent neural network for authorship attribution. *arXiv preprint arXiv:1902.09723*, 2019.

[50] N. Japkowicz and M. Shah. *Evaluating learning algorithms: a classification perspective*. Cambridge University Press, 2011.

[51] T. Joachims. A probabilistic analysis of the rocchio algorithm with tfidf for text categorization. Technical report, Carnegie-mellon univ pittsburgh pa dept of computer science, 1996.

[52] P. Juola and E. Stamatatos. Overview of the author identification task at pan 2013. *CLEF (Working Notes)*, 1179, 2013.

[53] D. Jurafsky and J. H. Martin. Speech and language processing (draft). *Chapter A: Hidden Markov Models (Draft of September 11, 2018). Retrieved March*, 19:2019, 2018.

[54] M. Kestemont, K. Luyckx, W. Daelemans, and T. Crombez. Cross-genre authorship verification using unmasking. *English Studies*, 93(3):340–356, 2012.

[55] M. Kestemont, E. Manjavacas, I. Markov, J. Bevendorff, M. Wiegmann, E. Stamatatos, M. Potthast, and B. Stein. Overview of the cross-domain authorship verification task at pan 2020. In *CLEF (Working Notes)*, 2020.

[56] M. Kestemont, M. Tschuggnall, E. Stamatatos, W. Daelemans, G. Specht, B. Stein, and M. Potthast. Overview of the author identification task at pan-2018: cross-domain authorship attribution and style change detection. In *Working Notes Papers of the CLEF 2018 Evaluation Labs. Avignon, France, September 10-14, 2018/Cappellato, Linda [edit.]; et al.*, pages 1–25, 2018.

[57] Y. Kim. Convolutional neural networks for sentence classification proceedings of the 2014 conference on empirical methods in natural language processing, emnlp 2014, october 25-29, 2014, doha, qatar, a meeting of sigdat, a special interest group of the acl. *Association for Computational Linguistics, Doha, Qatar*, 2014.

[58] T. N. Kipf and M. Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

[59] N. Kitaev and D. Klein. Constituency parsing with a self-attentive encoder. *arXiv preprint arXiv:1805.01052*, 2018.

[60] S. Konstantinou, A. Zinonos, and J. Li. Different encoding approaches for authorship verification. In *CEUR workshop proceedings*, volume 3180. CLEF, 2022.

[61] M. Koppel and J. Schler. Authorship verification as a one-class classification problem. In *Proceedings of the twenty-first international conference on Machine learning*, page 62, 2004.

[62] M. Koppel, J. Schler, and S. Argamon. Computational methods in authorship attribution. *Journal of the American Society for information Science and Technology*, 60(1):9–26, 2009.

[63] M. Koppel, J. Schler, and S. Argamon. Authorship attribution in the wild. *Language Resources and Evaluation*, 45:83–94, 2011.

[64] M. Koppel, J. Schler, and E. Bonchek-Dokow. Measuring differentiability: Unmasking pseudonymous authors. *Journal of Machine Learning Research*, 8(6), 2007.

[65] M. Koppel and Y. Winter. Determining if two documents are written by the same author. *Journal of the Association for Information Science and Technology*, 65(1):178–187, 2014.

[66] T. K. Landauer, P. W. Foltz, and D. Laham. An introduction to latent semantic analysis. *Discourse processes*, 25(2-3):259–284, 1998.

[67] Y. Li, R. Jin, and Y. Luo. Classifying relations in clinical narratives using segment graph convolutional and recurrent neural networks (seg-gcrns). *Journal of the American Medical Informatics Association*, 26(3):262–268, 2019.

[68] P. Liu, X. Qiu, and X. Huang. Recurrent neural network for text classification with multi-task learning. *arXiv preprint arXiv:1605.05101*, 2016.

[69] P. J. Liu, M. Saleh, E. Pot, B. Goodrich, R. Sepassi, L. Kaiser, and N. Shazeer. Generating wikipedia by summarizing long sequences. *arXiv preprint arXiv:1801.10198*, 2018.

[70] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

[71] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015.

[72] C. Manning and H. Schutze. *Foundations of statistical natural language processing*. MIT press, 1999.

[73] D. Marcheggiani and I. Titov. Encoding sentences with graph convolutional networks for semantic role labeling. *arXiv preprint arXiv:1703.04826*, 2017.

[74] M. P. Marcus, B. Santorini, and M. A. Marcinkiewicz. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330, 1993.

[75] Y. Marton, N. Wu, and L. Hellerstein. On compression-based text classification. In *Advances in Information Retrieval: 27th European Conference on IR Research, ECIR 2005, Santiago de Compostela, Spain, March 21-23, 2005. Proceedings 27*, pages 300–314. Springer, 2005.

[76] R. A. Matthews and T. V. Merriam. Neural computation in stylometry i: An application to the works of shakespeare and fletcher. *Literary and Linguistic computing*, 8(4):203–209, 1993.

[77] T. McEnery and A. Wilson. Corpus linguistics. *The Oxford handbook of computational linguistics*, pages 448–463, 2003.

[78] R. Mihalcea and D. Radev. *Graph-based Natural Language Processing and Information Retrieval*. Cambridge University Press, 2011.

[79] R. Mihalcea and D. Radev. *Graph-based natural language processing and information retrieval*. Cambridge university press, 2011.

[80] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. *Advances in neural information processing systems*, 26, 2013.

[81] J. Mothe, J. Savoy, J. Kamps, K. Pinel-Sauvagnat, G. Jones, E. Juan, and N. Ferro. Experimental ir meets multilinguality, multimodality, and interaction. In *6th International Conference of the CLEF Association (CLEF'15)*. Springer, 2015.

[82] M. P. Naeini, G. Cooper, and M. Hauskrecht. Obtaining well calibrated probabilities using bayesian binning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 29, 2015.

[83] M. Najafi and E. Tavan. Text-to-text transformer in authorship verification via stylistic and semantical analysis. In *Proceedings of the CLEF*, 2022.

[84] R. Navigli. Word sense disambiguation: A survey. *ACM computing surveys (CSUR)*, 41(2):1–69, 2009.

[85] V. M. Ngo and T. H. Cao. A generalized vector space model for ontology-based information retrieval. *arXiv preprint arXiv:1807.07779*, 2018.

[86] R. OpenAI. Gpt-4 technical report. *arXiv*, 2023.

[87] A. Peñas and A. Rodrigo. A simple measure to assess non-response. In *Proceedings of 49th Annual Meeting of the Association for Computational Linguistics - Human Language Technologies (ACL-HLT 2011).*, 2011.

[88] H. Peng, J. Li, Y. He, Y. Liu, M. Bao, L. Wang, Y. Song, and Q. Yang. Large-scale hierarchical text classification with recursively regularized deep graph-cnn. In *Proceedings of the 2018 world wide web conference*, pages 1063–1072, 2018.

[89] N. Potha and E. Stamatatos. Improving author verification based on topic modeling. *Journal of the Association for Information Science and Technology*, 70(10):1074–1088, 2019.

[90] D. M. Powers. Evaluation: from precision, recall and f-measure to roc, informedness, markedness and correlation. *arXiv preprint arXiv:2010.16061*, 2020.

[91] G. Salton and C. Buckley. Term-weighting approaches in automatic text retrieval. *Information processing & management*, 24(5):513–523, 1988.

[92] U. Sapkota, S. Bethard, M. Montes, and T. Solorio. Not all character n-grams are created equal: A study in authorship attribution. In *Proceedings of the 2015 conference of the North American chapter of the association for computational linguistics: Human language technologies*, pages 93–102, 2015.

[93] U. Sapkota, T. Solorio, M. Montes, S. Bethard, and P. Rosso. Cross-topic authorship attribution: Will out-of-topic data help? In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 1228–1237, 2014.

[94] N. Schaetti. Character-based convolutional neural network for style change detection. *Training*, 2980(1490):1490, 2018.

[95] S. S. Sonawane and P. A. Kulkarni. Graph based representation and analysis of text document: A survey of techniques. *International Journal of Computer Applications*, 96(19), 2014.

[96] E. Stamatatos. Intrinsic plagiarism detection using character n-gram profiles. *threshold*, 2(1,500), 2009.

[97] E. Stamatatos. A survey of modern authorship attribution methods. *Journal of the American Society for information Science and Technology*, 60(3):538–556, 2009.

[98] E. Stamatatos. Authorship verification: a review of recent advances. *Research in Computing Science*, 123:9–25, 2016.

[99] E. Stamatatos. Authorship attribution using text distortion. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 1138–1149, 2017.

[100] E. Stamatatos, W. Daelemans, B. Verhoeven, M. Potthast, B. Stein, P. Juola, M. A. Sanchez-Perez, and A. Barrón-Cedeño. Overview of the author identification task at pan 2014. In *CLEF 2014 Evaluation Labs and Workshop Working Notes Papers, Sheffield, UK, 2014*, pages 1–21, 2014.

[101] E. Stamatatos, M. Kestemont, K. Kredens, P. Pezik, A. Heini, J. Bevendorff, B. Stein, and M. Potthast. Overview of the authorship verification task at pan 2022. In *CEUR workshop proceedings*, volume 3180, pages 2301–2313, 2022.

[102] E. Stamatatos, M. Potthast, F. Rangel, P. Rosso, and B. Stein. Overview of the pan/clef 2015 evaluation lab. In *Experimental IR Meets Multilinguality, Multimodality, and Interaction: 6th International Conference of the CLEF Association, CLEF'15, Toulouse, France, September 8-11, 2015, Proceedings 6*, pages 518–538. Springer, 2015.

[103] P. D. Stamatatos et al. On the robustness of authorship attribution based on character n-gram features. *Journal of Law and Policy*, 21(2):7, 2013.

[104] C. Sun, X. Qiu, Y. Xu, and X. Huang. How to fine-tune bert for text classification? In *Chinese Computational Linguistics: 18th China National Conference, CCL 2019, Kunming, China, October 18–20, 2019, Proceedings 18*, pages 194–206. Springer, 2019.

[105] K. S. Tai, R. Socher, and C. D. Manning. Improved semantic representations from tree-structured long short-term memory networks. *arXiv preprint arXiv:1503.00075*, 2015.

[106] W. J. Teahan. *Modelling english text*. PhD thesis, University of Waikato, 1998.

[107] W. J. Teahan and D. J. Harper. Using compression-based language models for text categorization. *Language modeling for information retrieval*, pages 141–165, 2003.

[108] M. Tschuggnall and G. Specht. Countering plagiarism by exposing irregularities in authors' grammar. In *2013 European Intelligence and Security Informatics Conference*, pages 15–22. IEEE, 2013.

[109] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[110] D. Vilariño, D. Pinto, H. Gómez, S. León, and E. Castillo. Lexical-syntactic and graph-based features for authorship verification. In *Proceedings of CLEF*, pages 282–302, 2013.

[111] Z. Wang, J. Zhang, J. Feng, and Z. Chen. Knowledge graph embedding by translating on hyperplanes. In *Proceedings of the AAAI conference on artificial intelligence*, volume 28, 2014.

[112] C. Xiong, R. Power, and J. Callan. Explicit semantic ranking for academic search via knowledge graph embedding. In *Proceedings of the 26th international conference on world wide web*, pages 1271–1279, 2017.

[113] Z. Yang, W. Cohen, and R. Salakhudinov. Revisiting semi-supervised learning with graph embeddings. In *International conference on machine learning*, pages 40–48. PMLR, 2016.

[114] Z. Yang, Z. Dai, Y. Yang, J. Carbonell, R. Salakhutdinov, and X. Le QV. generalized autoregressive pretraining for language understanding; 2019. *Preprint at https://arxiv.org/abs/1906.08237 Accessed June*, 21, 2021.

[115] L. Yao, C. Mao, and Y. Luo. Graph convolutional networks for text classification. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pages 7370–7377, 2019.

[116] W.-t. Yih, X. He, and C. Meek. Semantic parsing for single-relation question answering. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 643–648, 2014.

[117] Y. Zhao and J. Zobel. Entropy-based authorship search in large document collections. In *Advances in Information Retrieval: 29th European Conference on IR Research, ECIR 2007, Rome, Italy, April 2-5, 2007. Proceedings 29*, pages 381–392. Springer, 2007.

[118] R. Zheng, J. Li, H. Chen, and Z. Huang. A framework for authorship identification of online messages: Writing-style features and classification techniques. *Journal of the American society for information science and technology*, 57(3):378–393, 2006.