Universidad Nacional Autónoma de México
Posgrado en Ciencias e Ingeniería de la Computación

VERIFICACIÓN DE AUTORÍA EN TEXTOS MEDIANTE REDES NEURONALES CONVOLUCIONALES Y RECURRENTES

TESIS
PARA OPTAR POR EL GRADO DE:
MAESTRO EN CIENCIAS E INGENIERÍA DE LA COMPUTACIÓN

P R E S E N T A:

F E R N A N D O L Ó P E Z V E L A S C O

Director de Tesis:
Dr. Demetrio Fabián García Nocetti
IIMAS, UNAM

Codirector de Tesis:
Dr. Gibrán Fuentes Pineda
IIMAS, UNAM

Ciudad Universitaria, CD. MX, Enero 2018

Resumen

El siguiente trabajo de tesis presenta una propuesta para resolver el problema de verificación de autor en base al estilo de escritura de cada uno de estos. El modelo que se propone se basa en una arquitectura siamés compuesta por redes neuronales convolucionales y recurrente de memoria de corto a largo plazo, el beneficio de la arquitectura siamés se ve reflejado en la reducción de pesos a ajustar en el modelo. Asimismo, se demuestra que para aprender el estilo de escritura de un autor no es suficiente tratar los textos a verificar con modelos de espacio vectorial. Para verificar la generalidad del modelo propuesto, se realizaron pruebas con dos conjuntos de datos externas las cuales fueron extraídos de la competencia PAN CLEF de los años 2013 y 2014. Los resultados obtenidos por el modelo que se propone en este trabajo de tesis están apegados al estado del arte.

Agradecimientos

Quiero agradecer de manera especial al programa de Posgrado en Ciencias e Ingeniería de la Computación, por haberme brindado un apoyo incondicional en toda mi estancia de estudios de maestría. Quiero agradecer a mis profesores, quienes con su conocimiento, me brindaron las herramientas intelectuales para poder superar los retos que en este camino me fui encontrando.

Por otra parte, quiero agradecer a mi director de tesis el Dr. Fabián García Nocetti por el apoyo brindado a lo largo del desarrollo de este trabajo. Asimismo, quiero dar las gracias a mi cotutor el Dr. Gibrán Fuentes Pineda, quien siempre estuvo disponible para escucharme, orientarme, aconsejarme y guiarme en este arduo camino; quien fue la persona que con su conocimiento, experiencia, pasión y dedicación, me motivó e inspiró a tomar este camino, el camino de la investigación.

Agradezco de manera especial a cada uno de mis sinodales a la Dra. Ma. Elena Lárraga Ramírez, al Dr. Carlos Gershenson García y al Dr. Iván Vladimir Meza Ruíz por sus comentarios y sugerencias para realizar este trabajo de tesis.

Quiero agradecer a mi novia, Lauris. Quien siempre me ha mostrado un apoyo incondicional en este camino que decidí tomar, quien ha sido un impulso y motivación constante desde el día en que decidí entrar al programa. Agradezco a mis padres por haberme formado como una persona capaz de siempre luchar por sus sueños. Agradezco a mi amigo Etan por su motivación y apoyo en la recolección de documentos.

Investigación realizada gracias al Programa de Apoyo a Proyectos de Investigación e Innovación Tecnológica (PAPIIT) de la UNAM IA104016 Generación de resúmenes de video basado en redes neuronales profundas de principio a fin.

Dedicatoria

Este trabajo lo dedico a quien siempre creyó en mi, a quien cuando estuve a punto de rendirme en este camino, me levantó. A quien ha sido el refuerzo en cada decisión que he tomado. A quien ha sido mi consiente cuando me he visto despegado de la realidad, a quien que de todas las maneras posibles siempre encuentra la manera de expresarme que soy el mejor. Este trabajo te lo de dedico a ti, quien en mis frustraciones, desesperaciones y tristezas, nunca me soltó. Este trabajo te lo dedico a ti Lauris, mi koala.

"For the past 33 years, I have looked in the mirror every morning and asked myself: 'If today were the last day of my life, would I want to do what I am about to do today?' And whenever the answer has been 'No' for too many days in a row, I know I need to change something."

Steve Jobs

Índice general

Índice de tablas

Índice de figuras

Capítulo 1

Introducción

Hoy en día, gracias a la facilidad con la que podemos hacer uso de plataformas sociales, es posible crear y compartir contenido tanto en formato de audio o video así como gran cantidad de documentos de texto, de una manera muy sencilla. Sin embargo, tal facilidad con la que se puede generar contenido nos lleva a escenarios en donde se tiene que analizar muy cuidadosamente la originalidad, veracidad o intencionalidad con la que tal contenido fue creado, compartido o modificado.

Desde hace varias décadas se han presentando casos en donde descifrar la veracidad, originalidad, intencionalidad o autenticidad de algún documento ha sido de vital importancia. Tal es el caso de los escritos federalistas [28] en donde se pretendió determinar la autoría de dichos escritos. Asimismo, se han detectado una gran cantidad de casos de plagio de diferente índole, dichos casos abarcan desde ensayos de estudiantes universitarios [1], hasta trabajos de tesis [2]. No obstante, grandes esfuerzos se han realizado por parte de especialistas en el área para poder desarrollar técnicas que den solución de manera eficiente a los problemas que

[1] http://www.bbc.com/news/uk-wales-36828071
[2] http://www.bbc.com/news/world-latin-america-37153951

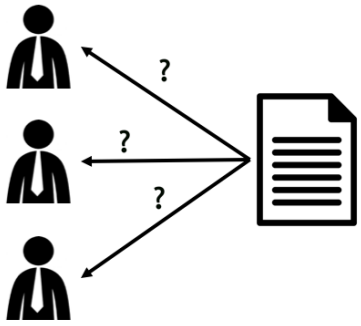

Figura 1.1: Diagrama de la tarea de identificación de autor.

involucran la autoría de documentos.

El análisis de autoría en textos en las últimas décadas ha tomado mucha mayor fuerza y popularidad. Existen diferentes tareas relacionadas al análisis de autoría tales como identificación de género, perfilado, identificación de personalidad, imitación, enmascaramiento, identificación, verificación y agrupación de autor. [3]. A continuación se describen las 3 tareas mas relevantes para este trabajo.

- **El problema de identificación de autor** refiere a, dado un conjunto de autores conocidos y un documento del cual no se conoce el autor, identificar el autor del documento en cuestión. La figura 1.1 describe de manera gráfica el problema de identificación de autor en donde se conocen los autores A, B y C y existe un documento x del cual se desconoce su autor. Se pretende identificar a qué autor de los conocidos pertenece dicho documento.

- **El problema de verificación de autor** refiere a, dado un conjunto de documentos de los cuales se sabe que son escritos por el mismo autor y dado un documento desconocido, verificar si el texto desconocido pertenece al autor o no. Este problema se describe de manera gráfica en la figura 1.2 en donde suponemos que tenemos al autor A, del cual

[3] http://pan.webis.de/tasks.html

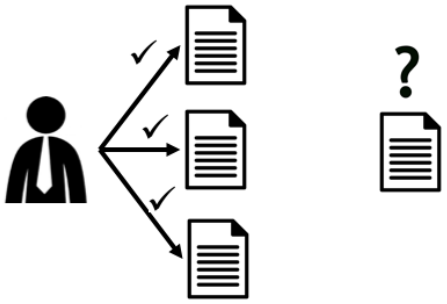

Figura 1.2: Diagrama de la tarea de verificación de autor.

conocemos sus documentos d_1, d_2 y d_3. Por otra parte tenemos al documento d_x del cual desconocemos su autoría. La tarea pretende verificar si el documento d_x pertenece o no al autor A.

- **El problema de agrupamiento por autor** es el problema de agrupar una serie de documentos por autor sin tener información sobre los autores o la cantidad de autores candidatos. La figura 1.3 describe de manera gráfica el problema de agrupamiento de autor en donde tenemos un conjunto de documentos d_1, d_2, d_3, d_4 y d_5, de los cuales no conocemos el autor de ninguno. La tarea consiste en agrupar aquellos documentos que pudieran pertenecer al mismo autor.

Tratar de determinar quién escribió un documento es una tarea sumamente compleja. Existe una cantidad muy basta de factores que determinan la originalidad de un documento dado. Tales como el tiempo verbal en que el autor redacta, la cantidad signos de puntuación o la frecuencia con la que hace uso de dichos signos [4], las palabras que más veces repite así como las que muy pocas veces utilizó, si el autor hace uso más frecuente de voz pasiva o de voz activa [5], etc.

La estilometría, la cual se refiere al análisis estadístico del estilo lingüístico de cierto autor

[4] http://www.learnnc.org/lp/editions/few/684
[5] https://letterpile.com/writing/Four-Types-of-Writing

Figura 1.3: Diagrama de la tarea de agrupamiento de documentos.

sobre un documento dado, es una técnica que se ha implementado comúnmente para resolver problemas de análisis de textos. [7] La estilometría computacional ha tomado un papel importante en tareas de distintos campos tales como la psicología, la lingüística forense, la socio-lingüística, leyes, periodismo y medicina. Dichas tareas, en un enfoque general, tratan de describir el comportamiento el autor en base al estilo en que redacta una carta, un artículo, un libro, un diagnóstico médico, etc [38].

En éste trabajo de tesis nos enfocaremos en el problema de verificación de autor haciendo énfasis en un caso muy particular, esto es, descifrar si dados dos documentos ambos son escritos o no por el mismo autor.

1.1. Planteamiento del problema

Dado un par de documentos, cada uno redactado por un único autor en el idioma inglés, con género y longitud del documento variable, se pretende encontrar si fueron escritos por el mismo autor. El problema de verificación de autor para un par de documentos, es un escenario común. Supongamos el caso en el que se cuenta con cierto documento a redactado por un autor X dado y otro texto b del cual se desconoce el autor. El autor X dice pertenecerle el documento sin autor conocido. La tarea es comparar el documento a con el b para

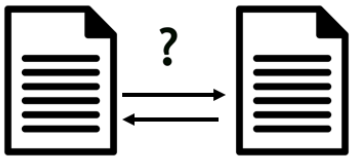

Figura 1.4: Diagrama de problema de verificación específico a resolver.

determinar si efectivamente ambos pertenecen o no al autor X. La figura 1.4 hace referencia a la comparativa que se debe realizar para determinar si un par de textos pertenecen o no a mismo autor.

1.2. Objetivo

Desarrollar un método que, en base al estilo de escritura pueda verificar si dados un par de documentos estos fueron redactados o no por el mismo autor.

1.2.1. Objetivos específicos

- Revisión del estado del arte referente a la verificación, identificación y agrupamiento de autor.
- Diseño y desarrollo de un modelo de espacio vectorial basado en árboles aleatorios que fungirá como línea de comparación base.
- Diseño y desarrollo de un arquitectura siamés basada en redes neuronales profundas para la verificación de autor.
- Evaluación del modelo de comparación base.
- Evaluación del modelo siamés basado en redes neuronales profundas

- Realizar variaciones en parámetros e hiperparámetros así como comparaciones entre el modelo de espacio vectorial y el modelo basado en redes neuronales profundas.

1.3. Metodología

Podemos tener un sistema que aprenda a verificar pares de documentos con base en el estilo de cada autor. Esto se puede lograr entrenando un modelo basado en una arquitectura siamés compuesta de un ensamble de redes convolucionales y recurrentes de memoria de corto a largo plazo a partir de un conjunto de datos con la cantidad suficiente de elementos tanto positivos como negativos.

Los modelos convencionales de espacio vectorial no mantienen el orden secuencial de un documento dado. Mantener el orden secuencial es de vital importancia para poder determinar el estilo de cada autor. Una solución para mantener el orden secuencial de cada palabra o carácter que componen a un documento, es la representación embebida.

El modelo que se propone y que está inspirado en una arquitectura siamés, está basado en un ensamble de redes convolucionales y recurrentes de memoria de corto a largo plazo, el cual es alimentado con secuencias de caracteres embebidos lo que nos garantiza mantener el orden secuencial de cada documento. Las redes convolucionales nos permiten abstraer características generales y representativas de cada secuencia, las cuales son alimentadas a la red recurrente de memoria de corto a largo plazo para capturar el estilo de cada autor.

1.4. Aportaciones

- Diseño y evaluación de una arquitectura siamés de aprendizaje profundo para el problema de verificación de autor con resultados comparables y competitivos con el estado del arte.
- Comparativa entre modelos de espacio vectorial convencionales y la arquitectura siamés de aprendizaje profundo.

1.5. Organización de la tesis

El presente trabajo de tesis está organizado de la siguiente manera:

- En el capítulo 2 se describen los trabajos para análisis de autoría, tales como los basados en distancias, en impostores y los basados en redes neuronales profundas.
- En el capítulo 3 se describen los métodos de espacio vectorial, así como las arquitecturas basadas en redes neuronales profundas.
- En el capítulo 4 se describe el método de espacio vectorial propuesto para fungir como línea base de comparación, así como la arquitectura siamés basada en redes neuronales convolucionales y recurrentes de memoria de corto a largo plazo para la verificación de autor.
- En el capítulo 5 se presentan los resultados experimentales del modelo de espacio vectorial así como de la arquitectura siamés propuesta.
- El capítulo 6 presentan las conclusiones de este trabajo de tesis. Asimismo, se sugieren posibles direcciones de trabajo futuro.

Capítulo 2

Antecedentes

A lo largo de este capítulo se describirán los trabajos realizados para el análisis de autoría así como las técnicas propuestas para poder resolver problemas de identificación, verificación y agrupamiento de autor. Los trabajos mencionados están agrupados de acuerdo a la semejanza en técnicas propuestas tales como, métodos basados en distancias, métodos basados en impostores y métodos basados en redes neuronales profundas.

2.1. Métodos basados en distancias

Uno de los enfoques para dar solución a los problemas de identificación, verificación o agrupación de autor, es haciendo uso de métricas de distancias. En la gran mayoría de los casos, en primera instancia se genera un modelo de espacio vectorial del cual se parte para aplicar alguna métrica. Algunas de las métricas más comunes son: la distancia Jaccard, Jaccard ponderado, Sorensen ponderado, Massi ponderado, distancia euclidiana y distancia del coseno [23].

Castro [5] presenta una propuesta para el problema de verificación realizando comparaciones entre las medidas de similitud de cada documento. Su método se basa en obtener el promedio de similitud de todos los documentos conocidos así como el promedio de similitud del documento no conocido respecto a los demás. De esta manera, si la similitud promedio del documento desconocido respecto a los demás es menor que la distancia promedio de los documentos conocidos, se concluye que dicho documento no pertenece al autor. Sin embargo, si la distancia promedio es mayor o igual al la distancia promedio de los documentos conocidos, se dice que el documento pertenece al autor.

Veenman [43] presenta un modelo para el problema de verificación de autor de la competencia PAN CLEF 2013[1]. Tal propuesta está dada como un modelo de optimización lineal en donde se pretende encontrar aquella región de decisión que se tome como referencia para decir si el documento pertenece o no al autor dado. La función objetivo del modelo busca minimizar la distancia del documento a verificar respecto a los documentos conocidos. Las restricciones por su parte están dadas por una combinación lineal de los vectores de pesos correspondientes a documentos fuera de contexto así como los documentos conocidos.

Petmanson [32] propone un método basado en la comparación de distancias euclidianas para dar solución a la tarea de identificación de autor de la competencia PAN CLEF 2013. La propuesta de Petmanson se basa en la extracción de características representativas de los documentos conocidos así como del documento no conocido para generar correlaciones a partir de las características representativas de cada documento y finalmente trasladarlos a un plano en donde compara las distancias haciendo uso de la distancia euclidiana para determinar la autoría del documento en cuestión.

[1] http://pan.webis.de/clef13/pan13-web/author-identification.html

2.2. Métodos basados en el método de impostores y variaciones

Otra alternativa que se ha propuesto para dar solución al problema de identificación de autor es el método de impostores propuesto por Koppel [20]. Su propuesta parte de la idea de que se puede resolver un problema de identificación de autor reduciéndolo a un problema de verificación de autor para un par de documentos. Partiendo de esta idea, propone generar documentos a los cuales llama impostores con la finalidad de tener una línea de comparación para determinar si dicho par de documentos pertenecen o no al mismo autor. Tal comparación consta en medir las distancias entre los vectores de características de cada uno de los documentos. De esta manera si la distancia es más corta entre los documentos conocidos se dice que pertenecen al mismo autor. Por otra parte, si la distancia del documento a verificar es más cercana a los documentos impostores, se dice que tal documento no pertenece al mismo autor.

Algorithm 1 MÉTODO DE IMPOSTORES

```
1: procedure VERIFICACIÓN(x, y, S)
2:     Input Dos documentos x y y. Un conjunto de impostores S
3:     p ← 0
4:     for k do
5:         tasa ← CaractersticasAleatorias()
6:         I ← ImpostoresAleatorios(S)
7:         if similitud(x, y) * similitud(y, x) > similitud(x, I_i) * similitud(y, I_i) then
8:             p ← p + 1/tasa
9:     if p > Δ* then
10:        return mismo                                   ▷ El mismo autor
11:    else
12:        return otro                                    ▷ Diferente autor
```

Seidman [37] propone una extensión al modelo basado en impostores propuesto por Koppel[19] para resolver el problema de identificación de autor para la competencia PAN CLEF 2013

[2]. Su propuesta plantea aplicar el método de impostores a cada par de documentos que se forman a partir del documento a identificar respecto a los documentos conocidos. De esta manera obtiene un promedio de distancias entre cada para de documentos. Si tal distancia promedio está por debajo del umbral, concluye que el documento no pertenece al autor, en otro caso, concluye que el documento pertenece al autor.

Algorithm 2 MÉTODO GENERAL DE IMPOSTORES

procedure IDENTIFICACIÓN(x, Y, S)
  Input x: Documento desconocido. y: Documentos conocidos. S: Impostores.
  for Y **do**
    $p[] \leftarrow Verificacion(x, y_i, S)$
  $\sigma \leftarrow promedio(p)$ ▷ Obtener promedio
  if $\sigma > \theta*$ **then**
    return *mismo* ▷ El mismo autor
  else
    return *otro* ▷ Diferente autor

Khonji [18] propone un trabajo que parte del método de Seidman [37]. Dicho trabajo es una extensión del método general de impostores el cual denomina ASGALF (*A slightly-modified GI-based Author-verifier with lots of features*). Su propuesta modifica las características que extrae de cada documento, esto debido a que incluye el tomar caracteres y palabras. Asimismo, varía la forma en el cálculo de distancias entre cada para de documentos.

Algorithm 3 MÉTODO ASGALF

procedure IDENTIFICATION(x, Y, S)
  Input x: Unknown document. y: Set of known authors. S: Set of impostors.
  for Y **do**
    $I \leftarrow Verification(x, y_i, S)$
  $p \leftarrow p + \frac{min-max(x,y)^2}{min-max(x,I_x)*min-max(y,I_y)}$ ▷ Donde I_x es el más cercano a x y I_y a y
  if $p >= 0$ and $p < 0.5$ **then**
    return *same* ▷ El mismo autor
  else if $p > 0.5$ and $p <= 1$ **then**
    return *other* ▷ Diferente autor
  else if $p = 0.5$ **then**
    return *unknown* ▷ No se sabe

[2]http://pan.webis.de/clef13/pan13-web/author-identification.html

2.3. Clasificadores bayesianos y máquinas de vectores de soporte

Moreau [26], plantea generar modelos de verificación de autor basados en algoritmos genéticos, a partir de los cuales genera un clasificador para determinar al mejor modelo, el cual complementa con un clasificador de máquinas de vectores de soporte para dar solución al problema de identificación de autor.

Posadas Durán [33] propone una metodología para dar solución al problema de perfilamiento de autor basado en el uso de modelos de espacio vectorial enfocada en n-gramas. Propone la implementación de un clasificador de máquinas de vectores de soporte para cada una de la etiquetas de perfil de autor en donde la intersección de cada uno de estos clasificadores determina el perfil del autor.

Sari [36] propone un modelo para identificación de género en documentos cortos. Su propuesta determina que la implementación de modelos de espacio vectorial basados en características que describen el estilo del autor en combinación con palabras función tienen un mejor desempeño al ser tratados con máquinas de soporte. En contraste, muestra que tales modelos de espacio vectorial tienen un bajo desempeño al ser tratados con modelos bayesianos.

Solorzano [40] propone una metodología para resolver el problema de verificación de autor para la competencia PAN CLEF basada en la extracción de distintos tipos de características las cuales son clasificadas mediante máquinas de vectores de soporte con distintas funciones de kernel.

Canales [4] desarrolló un sistema biométrico para autenticación de estudiantes basado en el estilo de escritura de cada estudiante. Su metodología se basa en el uso de modelos de espacio vectorial a nivel carácter así como de la implementación del algoritmo k-vecinos más

cercanos. Sus resultados mostraron ser competitivos y comparables con el estado del arte.

Mosteller y Wallace [28], propusieron una metodología para dar solución a un problema de identificación de autor. Su enfoque toma como base el uso modelos de espacio vectorial a nivel palabra así como la implementación de técnicas bayesianas para dar solución a dicho problema.

Iqbal [17] propone un método para resolver un problema de identificación de autor el cual aplica a un caso de detección de correos fraudulentos. Su propuesta se basa en la extracción de diferentes grupos de características las cuales combina para clasificar mediante máquinas de vectores de soporte y árboles de decisión.

2.4. Métodos basados en redes neuronales

Neculoiu [29] propone una arquitectura siamés para la detección de similitud en frases cortas. Parte de la idea de poder aprender las características representativas de cada frase para así compararlas mediante la distancia euclidiana y determinar qué tan similares son. Dicha arquitectura está compuesta por redes recurrentes de memoria de corto a largo plazo bidireccionales.

Dylan Rhodes [34] propone una metodología para resolver el problema de identificación de autor en el cual emplea redes neuronales convolucionales sobre los caracteres embebidos.

Kamil Safin [35] propone una metodología para la tarea de detección de anomalías en estilo para la competencia PAN CLEF 2017 [3]. Safin propone un método en el cual implementa un codificador-decodificador. Su método consiste en separar el documento en un conjunto de secuencias, las cuales las codifica a una espacio embebido. Posteriormente, utiliza dichas

[3]http://pan.webis.de/clef17/pan17-web/author-identification.html

secuencias embebidas para compararlas entre sí haciendo uso de la distancia del coseno. En su propuesta, hace variaciones al umbral para poder determinar entre qué par de secuencias hay una variación tal que determine que se ha detectado una anomalía y que por tanto se afirme que en esa parte del documento existe un cambio de estilo de autor.

Bagnall [2] propone una metodología para el problema de identificación de autor para la prueba de PAN CLEF 2015 [4]. Su propuesta parte del uso de redes neuronales recurrentes en un análisis a nivel carácter. La arquitectura propuesta hace uso de una red neuronal recurrente de multi-cabeza, es decir, que por cada estado arroja una salida. La función de activación utilizada en cada cabeza de cada estado de la red es una la función de raíz cuadrada desplazada rectificada ($ReSQRT$) para determinar la probabilidad de que pertenezca a algunos de los autores dados. Esta arquitectura fue probada para 4 diferentes idiomas, español, inglés, griego y holandés.

Shrestha [39] presentó un método de identificación de autor para secuencias cortas de mensajes en Twitter. El método que propone se basa en una arquitectura de redes neuronales convolucionales para un conjunto de datos de secuencias de 140 caracteres embebidos.

David R. [13] propone un modelo para el reconocimiento de similaridad de frases haciendo uso de redes neuronales convolucionales. Dicho trabajo experimenta con variaciones en la ventana del agrupamiento máximo para determinar la variación apropiada para su problema. La medida de comparación está basada en la implementación de la distancia cosénica así como la distancia euclidiana.

[4] http://pan.webis.de/clef15/pan15-web/author-identification.html

Capítulo 3

Fundamentos

3.1. Modelos de espacio vectorial

El modelo de espacio vectorial busca mapear cada término de cada documento en un índice lo permite contabilizar y obtener estadísticos para cada documento y el conjunto de datos. La indexación se puede aplicar a diferentes extracciones de un documento dado, es decir, se puede indexar un documento a nivel carácter, palabra, n-gramas, etc. Esta indexación permite construir un histograma a partir de un documento donde se ve reflejada la frecuencia de cada índice en el documento (a esto se le conoce como frecuencia de términos o **tf** por sus siglas en inglés).

Sin embargo, cuando se trabaja con conjuntos de datos compuesto por varios documentos, en donde se busca medir que tan relevante es una palabra para cierto documento, se aplica la frecuencias de documentos inversa (*idf* por sus sigla en inglés) la cual consiste en la contabilización de los N documentos que conforman el conjunto de datos y se divide entre el número de documentos en donde el término i aparece. La ecuación 3.1 describe la determinación de

las frecuencias de documentos inversa.

$$idf_i = \log\left(\frac{N}{df_i}\right) \tag{3.1}$$

Por otra parte, comúnmente se aplica la técnica conocida como frecuencia de términos - frecuencias de documentos inversa (*tf-idf* por sus siglas en inglés), la cual es una combinación del conteo de la frecuencia del término pesado con la frecuencia de documentos inversa.

3.1.1. Bosques aleatorios

Los bosques aleatorios son implementados para resolver tareas de clasificación y regresión los cuales están formados por ensambles de árboles de decisión. Los árboles de decisión son modelos intuitivos en los cuales a partir de la raíz se van generando bifurcaciones hasta cumplir con un criterio deseado. Generar ensambles de árboles o bosques aleatorios, tiene la ventaja de reducir la varianza debido a que cada resultado de cada ensamble es promediado.

Un bosque aleatorio es un clasificador que está compuesto por una colección de árboles de decisión. $\{h(x, \theta_k), k = 1, ..., n\}$ en donde $\{\theta_k\}$ son vectores aleatorios independientes e idénticamente distribuidos donde cada árbol vota por el vector de entrada x más popular.

Para un ensamble de clasificadores basados en árboles de decisión $\{h_1(x), h_2(x), ..., h_k(x)\}$ y con valores de entrenamiento aleatorios $\vec{y}$, el vector $\vec{x}$ define la función de margen como se muestra en la ecuación 3.2

$$margen(\vec{x}, \vec{y}) = (promedio_k) * I(h_k(\vec{x}) = \vec{y}) - max_{j \neq \vec{y}} * (promedio_k) * I(h_k(\vec{x}) = j) \tag{3.2}$$

en donde $I(.)$ es la función indicatriz. El margen tiene la función de tomar el promedio de los votos para la clase correcta Y dado X.

3.2. Redes neuronales

Una neurona artificial simple está compuesta por valores de entrada $\{x_1, x_2, ..., x_i\}$, pesos $\{w_1, w_2, ..., w_n\}$, un sesgo b, una función de activación y una salida u objetivo y. La estructura de una neurona artificial se muestra en la figura 3.1. Una neurona artificial está definida por una función base la cual está dada por la suma del producto de los pesos por los valores de entrada mas el sesgo como se observa en la ecuación 3.3. La función base es operada por una función de activación (ecuación 3.4), por ejemplo, la función sigmoide la cual se observa en la ecuación 3.5 o la función reLu la cual se observa en la ecuación 3.6.

Una composición de neuronas artificiales es denominada red neuronal artificial también conocida como red completamente conectada en la cual las neuronas artificiales que la componen están organizadas en capas como se observa en la figura 3.2. Una red neuronal artificial consta de 3 capas las cuales son, la capa de entrada, oculta y de salida. La capa de entrada representa a los valores que la red recibe, la capa oculta está definida por neuronales simples y la capa de salida representa a vector o vectores objetivo.

$$z = \sum_{i=0}^{n} (w_i x_i) + b \tag{3.3}$$

$$a = \sigma(z) \tag{3.4}$$

$$f(x) = \frac{1}{1 + e^{-x}} \tag{3.5}$$

$$f(x) = max\,(x, 0) \tag{3.6}$$

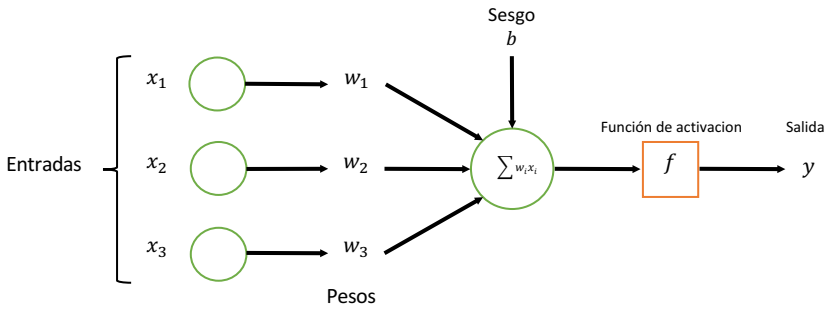

Figura 3.1: Estructura básica de una neurona artificial

3.2.1. Redes neuronales convolucionales

Las redes neuronales convolucionales [22] fueron desarrolladas principalmente para tareas relacionadas al reconocimiento de imágenes. Su arquitectura permite preservar la estructura espacial del problema [3]. Sin embargo, también se han aplicado a tareas de reconocimiento de voz y procesamiento del lenguaje natural, arrojando resultados que han contribuido al estado del arte.

El nombre convolucional fue dado debido al uso de la función matemática de convolución.

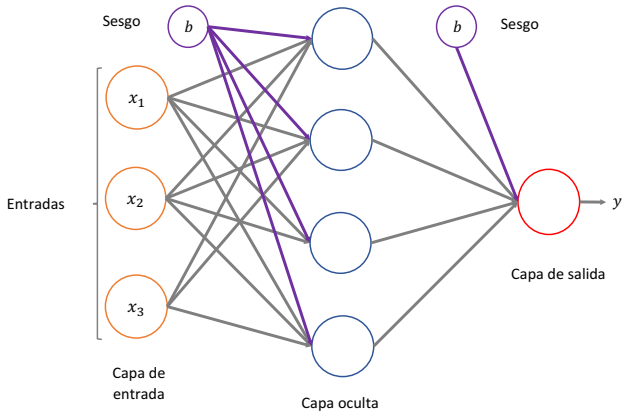

Figura 3.2: Red neuronal completamente conectada

La operación convolución es denotada por un asterisco como lo podemos ver en la ecuación 3.7. Las redes convolucionales utilizan la función convolución en lugar de un producto punto como sucede con un perceptrón multicapa.

$$s(t) = (x * w)(t). \tag{3.7}$$

En una red neuronal convolucional el argumento x refiere al vector de entrada mientras que la función w refiere al filtro. La salida es comúnmente conocida como un mapa de características. La ecuación 3.8 detalla al operador convolución en donde s(t) representa al mapa de características.

$$s(t) = (x * w)(t) = \sum_{a=-\infty}^{\infty} x(a)w(t-a). \tag{3.8}$$

Usualmente una red convolucional es alimentada imágenes. La ecuación 3.9 describe a un

mapa de características $S(i,j)$ dado por una imagen $I(i,j)$ así como un filtro de 2 dimensiones $K(i,j)$.

$$S(i,j) = (K * I)(i,j) = \sum_m \sum_n I(i-m, j-n)K(m,n). \tag{3.9}$$

En la figura 3.3 se describe la composición de una red neuronal convolucional. La figura es interpretada de izquierda a derecha en donde la entrada es una imagen a la cual se aplica la función de convolución. El filtro recorre la imagen de arriba a abajo. En cada punto el filtro genera un valor del mapa de características. Los elementos o valores que conforman el mapa de características se agrupan mediante una función de agrupamiento. Dicha función reducirá el tamaño del mapa de características tomando en cuenta el elemento más significativo de cada bloque o realizando un promedio según sea el caso. En algunos casos la capa de agrupamiento asocia sus valore a una capa de una red completamente conectada.

3.2.2. Redes neuronales recurrentes

Las redes neuronales recurrentes están orientadas a capturar relaciones secuenciales. Dichas redes constan de ciclos en su arquitectura lo que les permite tener una retroalimentación y memoria a través de tiempo. En la figura 3.5 podemos observar una arquitectura simple de una red recurrente.

Existen distintos tipos de arquitecturas en las redes neuronales recurrentes cada tipo de arquitectura está orientada a cierto problema. En la figura 3.4 se ilustran las taxonomías propuestas por Andrej Karpathy [3].

- **Una a muchas**: este tipo de arquitectura es aplicada a aquellas tareas en donde se busca describir un único elemento de entrada. Un ejemplo es la tarea de describir

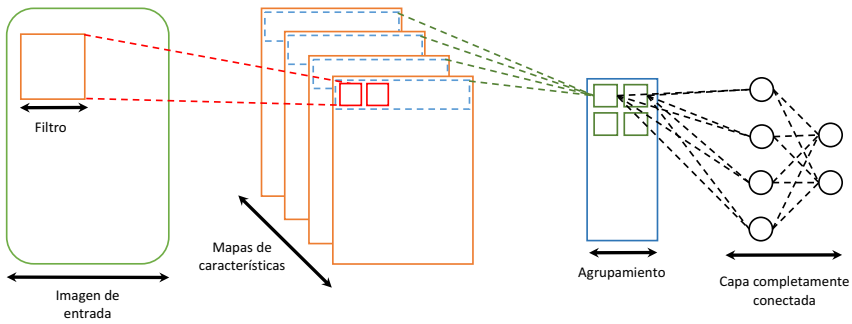

Figura 3.3: Ejemplo de una red neuronal convolucional

imágenes en donde se recibe una imagen como entrada y se produce una secuencia de palabras (la descripción textual) como salida.

- **Muchas a una**: este tipo de arquitectura recibe como entrada una secuencia de elementos mientras que la salida es una sólo vector que describe a la secuencia de entrada. Tales elementos de entrada pueden ser por ejemplo, secuencias de caracteres que describen alguna frase. Una de las tareas en donde este tipo de arquitecturas se aplican es en la clasificación de sentimientos.

- **Muchas a muchas**: este tipo de arquitectura recibe como entrada un conjunto de secuencias y la salida está dada por un conjunto de secuencias. Una de las tareas más comunes para este tipo de arquitecturas son los traductores, en donde se recibe una

secuencia de caracteres de entrada y de la misma manera, la salida está dada por otra secuencia de caracteres que representan a una frase.

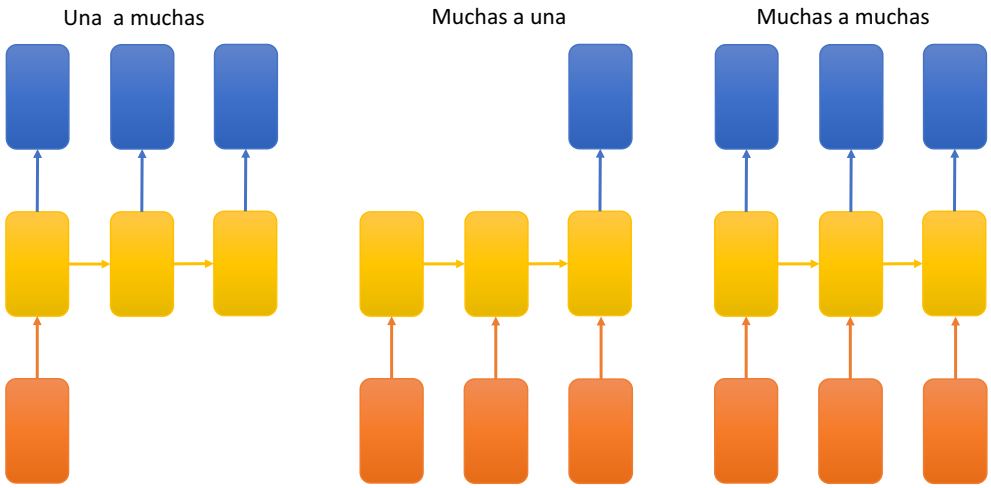

Figura 3.4: Tipos de arquitecturas de redes neuronales recurrentes

Las redes neuronales recurrentes, se pueden visualizar como una conexión a través del tiempo observar en la figura 3.6. Una red recurrente, al instante de tiempo t^i, es un perceptrón multicapa. La diferencia está muy marcada en que las redes recurrentes generan conexiones a través del tiempo.

Existen distintos tipos de algoritmos de aprendizaje para redes recurrentes. Un perceptrón multicapa puede aprender con el algoritmo de *propagación hacia atrás*. Si quisiéramos aplicar dicho algoritmo a una red recurrente, rompería en el instante en que tenga que evaluar

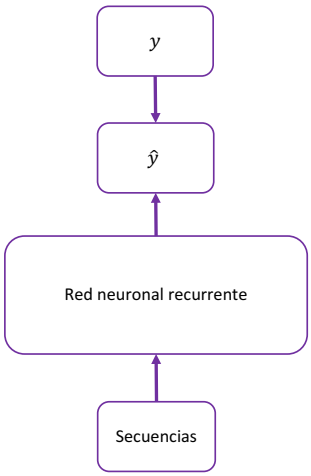

Figura 3.5: Red Neuronal Recurrente en su forma simple. Recibe un vector de entrada, para el ejemplo es un vector de secuencias, pasa por la recurrencia mientras que se calcula el valor de Y' haciendo uso de la etiqueta real que está dada por Y.

aquellas conexiones que generan ciclos o recurrencia. Para este problema se adaptó el algoritmo *propagación hacia atrás* para las redes recurrentes, llamado *Propagación hacia atrás a través del tiempo* (BPTT por sus siglas en inglés). Dicho algoritmo está adaptado para poder evaluar de manera correcta las conexiones recurrentes que se presentan en la arquitectura de una red recurrente.

Uno de los problemas que las redes recurrentes enfrentan es que, cuando se trabaja con una red profunda, es decir, que su arquitectura está constituida por varias capas, la red suele volverse inestable al momento del entrenamiento. Puede pasar que los pesos se vuelvan demasiado grandes o por el contrario, que los pesos tiendan a volverse demasiado pequeños.

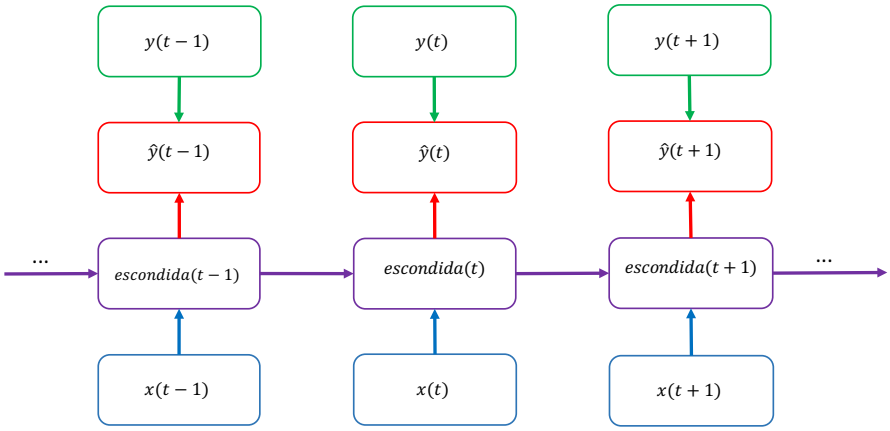

Figura 3.6: Red Neuronal Recurrente en su forma desplegada. La retroalimentación se realiza para cada instante de tiempo t.

En general, se dice que en la práctica las redes recurrentes convencionales no pueden capturar las relaciones a largo plazo. Sin embargo, un tipo de red recurrente conocida como Memoria de Corto y Largo Plazo (LSTM por sus siglas en inglés [15]) da solución a este tipo de problemas.

3.2.3. Red de memoria de corto a largo plazo

Las redes de memorias de corto y largo plazo resuelven el problema de la inestabilidad al trabajar con secuencias demasiado largas que las redes recurrentes convencionales enfrentan. Las LSTM son ideales para trabajar con grandes secuencias. Este tipo de redes recurrentes

son entrenadas usando el algoritmo *Propagación hacia atrás a través del tiempo*. Las redes recurrentes convencionales utilizan neuronas en cada una de sus capas (como ya se mencionó, pueden visualizarse como una cadena de redes perceptrón multicapa), sin embargo, las redes LSTM utilizan bloques que están conectados a través de las capas, en lugar de simples neuronas.

Las LSTM han sido muy exitosas en diferentes aplicaciones como por ejemplo, reconocimiento de escritura a mano sin restricciones [12], reconocimiento de voz [11], generación de texto a mano [10], entre otros. Están constituidas por celdas las cuales cuentan con compuertas que determinan y controlan el flujo de la información. Existen 3 tipos de compuertas [3]:

- **Compuerta de olvido**: Controla qué y cuanta información se mantiene en la celda
- **Compuerta de entrada**: Controla cuanta información nueva entra en la celda
- **Compuerta de salida**: Controla qué y cuanta información sale de la celda

Al igual que una red recurrente convencional, una LSTM tiene forma de cadena, es decir, cada celda que conforma a la red están vinculados a través del tiempo. Una de las partes esenciales de este tipo de redes es la celda ya que es en está en donde la red mantiene, desecha y recibe información. En primera instancia, la entrada x se concatena con el valor del estado anterior para pasar por la compuerta de olvido. Esto lo podemos visualizar en la figura 3.7. En una siguiente instancia, se modifica el estado de la celda, esto se logra pasado por una función sigmoidal así como función de tangente hiperbólica con la finalidad de obtener un vector candidato. Se aplica el producto punto entre el vector candidato y la salida de la función sigmoide para generar el vector del estado actual de la celda $c(t)$. Finalmente se calcula el vector de salida. Dicho vector parte del vector del estado actual de la célula $c(t)$ el cual pasa en una forma filtrada en un rango de 1 a -1. Esto se logra pasado el vector del estado de la

célula por la función de tangente hiperbólica, el resultado es operado mediante el producto punto con el vector de entrada x operado por una función sigmoide y que en efecto, generará el vector $h(t)$.

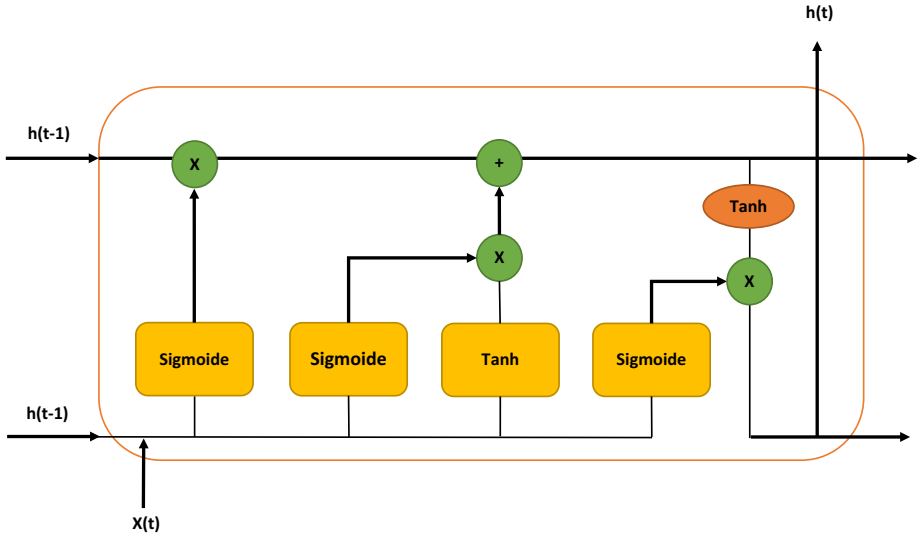

Figura 3.7: Memoria de corto y largo plazo. Esta figura fue inspirada en el artículo redactado por Olah [30]

Uno de los componentes más importantes es el estado de la célula, el cual denotaremos con $s_i^{(t)}$. Asimismo, uno de los elemento que influyen dentro de dicha célula es la compuerta de olvido o *forget gate* denotada por $f_i^{(t)}$ para el tiempo t en la celda i. Dicha función está dada por:

$$f_i^{(t)} = \sigma\left(b_i^f + \sum_j U_{i,j}^f x_j^{(t)} + \sum_j W_{i,j} h_j^{(t-1)}\right) \tag{3.10}$$

en donde $x^{(t)}$ representa al vector de entrada, $h^{(t)}$ representa a la celda oculta. b^f, U^f y W^f representan la entrada del vector sesgo, el vector de pesos de la capa de entrada y los pesos de las conexiones recurrentes, respectivamente. El estado de cada celda, $s^{(t)}$ se modifica de la siguiente manera:

$$s_i^{(t)} = f_i^{(t)} s_i^{(t-1)} + g_i^{(t)} \sigma \left({}_i + \sum_j U_{i,j} x_j^{(t)} + \sum_j W_{i,j} h_j^{(t-1)} \right) \tag{3.11}$$

en donde b, U y W representan al vector sesgo de entrada, los pesos de la capa de entrada y los pesos de las capas recurrentes respectivamente.

3.3. RMSProp

No es conveniente realizar el entrenamiento de una red neuronal recurrente mediante el uso únicamente de mini-lotes o de hiperparámetros de momento, debido a que la convergencia puede tomar largos periodos. La función de optimización RMSProp [14, 42] (*Root Mean Squared Error* por sus siglas en inglés) es la alternativa para este tipo de problemas. La idea de RMSProp es mantener un promedio móvil del cuadrado de los gradientes que son adyacentes de cada mini-lote con la finalidad de acelerar la convergencia. En el optimizador RMSProp la tasa de aprendizaje se adapta a cada uno de los vectores de pesos [21]:

$$\vec{v}(c_j^{(l)}, t) = \gamma \vec{v}(c_j^{(l)}, t-1) + (1-\gamma)(\partial J / \partial c_j^{(l)})) \tag{3.12}$$

$$\vec{v}(r_j^{(l)}, t) = \gamma \vec{v}(r_j^{(l)}, t-1) + (1-\gamma)(\partial J / \partial r_j^{(l)})) \tag{3.13}$$

de las cuales el valor de γ es el factor de pérdida. Posteriormente, la tasa de aprendizaje de cada vector de pesos es dividida por el promedio acumulado como se muestra en las ecuaciones

3.14 y 3.15 respectivamente:

$$c_j^{(l)} := c_j^{(l)} - \frac{\eta}{\sqrt{\vec{v}(c_j^{(l)})}} \cdot (\partial J / \partial c_j^{(l)}) \tag{3.14}$$

$$r_j^{(l)} := r_j^{(l)} - \frac{\eta}{\sqrt{\vec{v}(r_j^{(l)})}} \cdot (\partial J / \partial r_j^{(l)}) \tag{3.15}$$

3.4. Distancia coseno y distancia euclidiana

Dos de las métricas más utilizadas para análisis de autoría en textos, son la distancia euclidiana y la distancia del coseno. La implementación de alguna de estas métricas está dada por la finalidad de la tarea.

Se ha demostrado que la distancia coseno tiene un mejor desempeño en tareas en donde se da prioridad a la frecuencia de ocurrencia de cada palabra o carácter en todo el conjunto de datos. Por ejemplo, una de las tareas en donde la distancia coseno es aplicada como medida de similitud es en análisis de sentimientos [1]. La ecuación 3.16 define la distancia coseno.

En contraste, [1]la distancia euclidiana 3.17 tiene un mejor desempeño en tareas en donde se busca un parámetro de similitud basado en la sintáctica o semántica de dos vectores. Por ejemplo, unas de las tareas que implementan este tipo de métrica de similitud son identificación y verificación de autor en cuyas tareas se busca realizar una comparación en base a estilos. La ecuación 3.17 define la distancia euclidiana.

$$coseno(\vec{x}, \vec{y}) = \frac{\vec{x} \cdot \vec{y}}{\| \vec{x} \| \| \vec{y} \|} \tag{3.16}$$

[1]https://cmry.github.io/notes/euclidean-v-cosine

$$euclidiana(\vec{x}, \vec{y}) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2} \tag{3.17}$$

3.5. Error cuadrático medio

Una de las funciones para medir la pérdida en el entrenamiento de modelos para problemas de clasificación y regresión es la función del error cuadrático medio (ECM). Dicha función devuelve el promedio de las diferencias entre el valor que se busca modelar y el valor estimado al cuadrado. Básicamente el resultado de aplicar dicha función nos da una perspectiva de qué tanto fue la variación de lo que se esperaba a lo que el modelo arroja. Mientras más cercano a cero, es mejor. La función del error cuadrático medio la podemos observar en la ecuación 3.18

$$ECM = \frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2 \tag{3.18}$$

3.6. *Dropout*

El sobre-ajuste es un gran problema que se presenta cuando se trabaja con redes neuronales profundas con una gran cantidad de pesos a ajustar. El problema se presenta por varias razones las cuales pueden ser que el conjunto de datos no es lo suficientemente representativo al fenómeno que se quiere modelar o simplemente la red es demasiado profunda lo que genera que los parámetros se adapten al conjunto de datos.

Srivastava et al.[41] propusieron una técnica que ayuda a reducir el sobre-ajuste más conocida como *dropout*. La técnica del *dropout* consiste en deshabilitar temporalmente neuronas de la red junto con sus conexiones. La decisión de qué neuronas deshabilitar se lleva a cabo de manera aleatoria partiendo de una probabilidad p para cada neurona. En la figura 3.8 se

muestra el comportamiento de una arquitectura de red neuronal antes de aplicar *dropout* así como la misma arquitectura después de aplicar *dropout*. Como se observa en la imagen, tanto la neurona como sus conexiones son deshabilitadas.

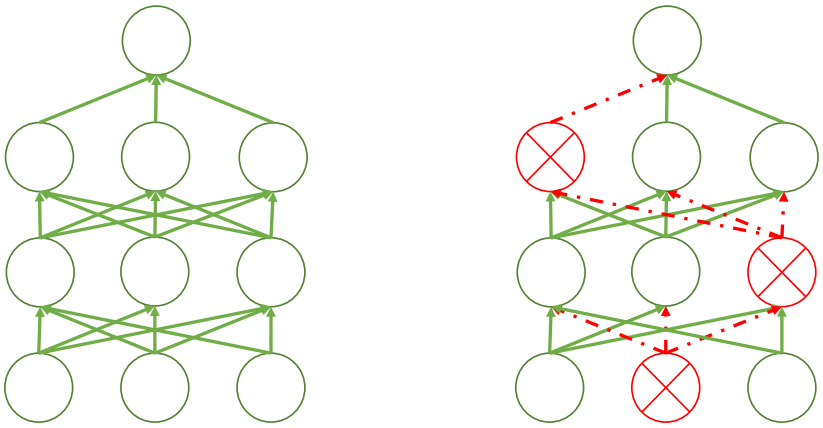

Figura 3.8: Descripción gráfica de la aplicación de *dropout* a una red neuronal. De lado izquierdo se observa un perceptrón multicapa estándar. De lado derecho se observa el mismo perceptrón con la aplicación del *dropout*, de color rojo se muestran aquellas neuronas que son deshabilitadas junto con sus conexiones.

3.7. Normalización por lotes

Uno de los inconvenientes que existe al emplear modelos de redes neuronales profundas es que el entrenamiento tiende a ser lento en relación a la dimensión de los de los elementos de

entrada así como la cantidad de pesos a ajustar. Una de las causas de esto es el cambio de distribución entre cada capa que conforma a la red neuronal lo que lleva a tener una tasa de aprendizaje bajo que en efecto hace que entrenar la red se vuelva más complicado. Dicho cambio de distribución en la red neuronal se conoce como *cambio covariable interno.*

Ioffe et al.[16] propusieron una técnica para acelerar el entrenamiento de una red neuronal profunda conocida como normalización por lotes. Dicha técnica busca eliminar el cambio covariable interno lo que en efecto estandariza las distribuciones obtenidas en cada capa de la red y permite emplear un valor de tasa de aprendizaje alto que en consecuencia genera un efecto de aceleración en el entrenamiento de la red.

El algoritmo 4 describe el procedimiento general de la normalización por lotes. La normalización para cada capa de la red neuronal se realiza en cada mini-lote en donde se toma cada elemento, se calcula la media, la varianza y posteriormente se normaliza el mini-lote.

Algorithm 4 NORMALIZACIÓN POR LOTES

procedure NORMALIZACIÓN(γ, β)
  Entrada Valores de x del mini lote $\mathcal{B} = \{x_1, ..., m\}$. Parámetros a ser aprendidos γ, β
  Salida $\{y_i = NL_{\beta,\gamma}(x_i)\}$
  $\mu_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i$ ▷ Promedio del mini lote
  $\sigma^2_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu_{\mathcal{B}})^2$ ▷ Varianza del mini lote
  $\hat{x_i} = \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma^2_{\mathcal{B}} + \epsilon}}$ ▷ Normalización
  $y_i \leftarrow \gamma\hat{x_i} + \beta \equiv NL_{\gamma,\beta}(x_i)$ ▷ Escalamiento y cambio

Capítulo 4

Verificación de autoría en textos

En este capítulo mostraremos y desarrollaremos las arquitecturas propuestas para la verificación de autor basadas en modelos de espacio vectorial así como en redes neuronales profundas. El capítulo está divido en dos secciones, la primer sección aborda una explicación detallada de cada modelo de espacio vectorial propuesto, la segunda sección detallará el modelo propuesto basado en redes neuronales profundas.

4.1. Verificación de autoría basado en bosques aleatorios

El clasificador basado en bosques aleatorios se ha implementado para resolver problemas de análisis de autoría como la propuesta realizada por Maitra [24] quien presentó una arquitectura basada en bosques aleatorios para resolver la tarea de verificación de autor para la competencia PAN CLEF 2016. La arquitectura de bosques aleatorios que presentamos como modelo de comparación base, utiliza 15 árboles de decisión, para cada uno de estos se implementa el criterio Gini [9] para medir la pureza de cada bifurcación realizada para cada árbol. El criterio Gini está dado por:

$$Gini(t) = 1 - \sum_{i=0}^{c-1}[p(i|t)]^2 \tag{4.1}$$

en donde t es el nodo a bifurcar y cada elemento i representa a cada atributo del nodo. Cada bifurcación se realiza a partir de dos elementos y se toma como hoja del árbol a partir de 1 elemento que conforme el árbol. En la figura 4.1 se describe la arquitectura del clasificador implementado para la verificación de autoría haciendo uso de bolsas de palabras de n-gramas de caracteres y palabras.

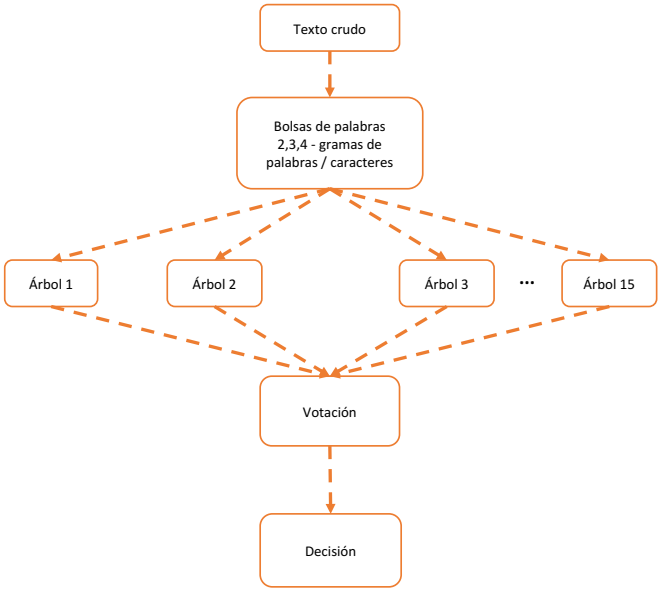

Figura 4.1: Clasificador de bosques aleatorios para verificación de autoría a partir de bolsas de palabras de n-gramas.

4.2. Verificación de autoría mediante redes neuronales

La arquitectura basada en redes neuronales profundas propuesta en este trabajo de tesis, fue inspirada en la arquitectura propuesta por Chopra et al.[6] para su trabajo sobre verificación de rostros y en la arquitectura propuesta por Necoloiu et al.[29] para su trabajo sobre similitud en frases, los cuales implementan una arquitectura *siamés*. Chopra et al. propusieron la arquitectura siamés para resolver una tarea enfocada a verificación de rostros en donde se pretende encontrar una función tal que al recibir un par de imágenes de rostros, arrojara si pertenecían o no a la misma persona. Necoloiu et al. por su parte, implementaron la idea de la arquitectura siamés para resolver la tarea de similitud de frases en donde se pretende determinar si dadas dos frases, estas tienen o no el mismo significado semántico.

La arquitectura siamés tiene la particularidad de estar formada por bloques de redes neuronales cuyos pesos son compartidos, es decir, al recibir dos elementos de entrada cada uno de estos actualiza los pesos de la red respectivamente. Generalmente la salida correspondiente a cada una de las entradas pasa por una función de energía basada en una métrica de distancia, por ejemplo la distancia euclidiana o la distancia del coseno. Una de las ventajas de esta arquitectura es que los pesos a actualizar son reducidos en cantidad lo cual favorece a evitar el sobre-ajuste del modelo.

La arquitectura que presentamos trabaja con secuencias de caracteres embebidos. Los caracteres embebidos son una extensión de lo que es conocido como palabras embebidas. Uno de los trabajos más importantes referentes a las palabras embebidas es el desarrollado por Mikolov et al.[25] en el cual de forma detallada explican que se puede llegar a entrenar un modelo tal que describa el comportamiento de una palabra dado un contexto en base a la relación con palabras vecinas en una ventana de análisis para cada palabra. Es decir, describir una palabra en base a la co-ocurrencia de dicha palabra con relación a la ocurrencia con

palabras en un contexto dado. Describir palabras basadas en la ocurrencia con otras palabras, tiene un gran potencial en donde se cuenta con una longitud adecuada de secuencias de palabras, pero no así con frases que son muy cortas o simplemente cuando lo que se busca no solo es encontrar una relación entre palabras sino una relación entre los caracteres de las palabras. Si se desea describir el comportamiento del estilo de un autor, se requiere conocer factores mucho más específicos que están fuertemente relacionados con la co-ocurrencia de los símbolos y caracteres que el autor utiliza al escribir.

La arquitectura que proponemos recibe como entrada dos vectores de características embebidos [27]. La representación de las secuencias en vectores embebidos es implementada debido a que al trabajar con palabras o caracteres, estamos hablando de datos categóricos, los cuales no son adecuados para con redes neuronales [8]. Un vector embebido nos provee de una representación tal que describe el comportamiento del elemento en cuestión, ya sea carácter o palabra, en relación a la co-ocurrencia con los demás elementos de la secuencia. Cada entrada al modelo corresponde a una secuencia de vectores embebidos que representa a un texto de un autor dado.

4.2.1. Ensamble de capas convolucionales y recurrente

La arquitectura siamés propuesta está constituida por un ensamble de dos capas de redes neuronales convolucionales y una red recurrente LSTM. El ensamble doble de capas convolucionales permite abstraer características locales a nivel carácter de cada secuencia, es decir, el ensamble de dos capas convolucionales permite abstraer características representativas de segmentos cortos de cada secuencia de caracteres. Tales características capturadas por las redes convolucionales son agrupadas mediante la técnica de agrupamiento máximo. Esta técnica permite resumir las características capturadas por las redes convolucionales, es decir, se toman aquellas características que representan de manera significativa a la secuencia

completa a partir de las características capturas por segmentos. Estas últimas características pasan por una memoria de corto y largo plazo en la cual captura las características globales del estilo de cada secuencia a partir de las características locales obtenidas por las capas convolucionales. Finalmente, la red recurrente LSTM arroja un vector para cada secuencia de entrada en donde la métrica de comparación está dada por la distancia euclidiana para cada par de secuencias.

En la figura 4.2 se muestran la arquitectura siamés basada en un ensamble de redes convolucionales y una recurrente de memoria de corto y largo plazo. En la figura 4.3 se ilustra la funcionalidad de la arquitectura siamés la cual recibe dos secuencias de textos en forma de vectores embebidos. Estos vectores pasan por una doble capa de redes neuronales convolucionales seguido del agrupamiento máximo cuya salida entra en forma de secuencia a una memoria de corto y largo plazo la cual arroja un vector de características globales de estilo para cada secuencia. Finalmente, estos dos vectores obtenidos es comparan con una distancia euclidiana que cuantifica la diferencia en estilo de las dos secuencias.

En la tabla 4.1 se describen los argumentos que componen cada una de las capas del la arquitectura siamés.

Capa	Número de filtros	Tamaño del filtro	Desplazmiento	Ventana de agrupamiento	Activación	Inicializador de kernel	Células ocultas	Dropout	Dropout recurrente
CNN 1D	75	15	1	-	Relu	Uniforme Glorot	-	0.1	-
CNN 1D	50	12	1	-	Relu	Uniforme Glorot	-	0.1	-
Agrupamiento 1D	75	12	1	4,6,8	Relu	Uniforme Glorot	-	0.1	-
LSTM	-	-	-	-	Relu	Uniforme Glorot	64	0.1	0.1

Tabla 4.1: Descripción de parámetros para cada una de las capas que componen a la arquitectura siamés

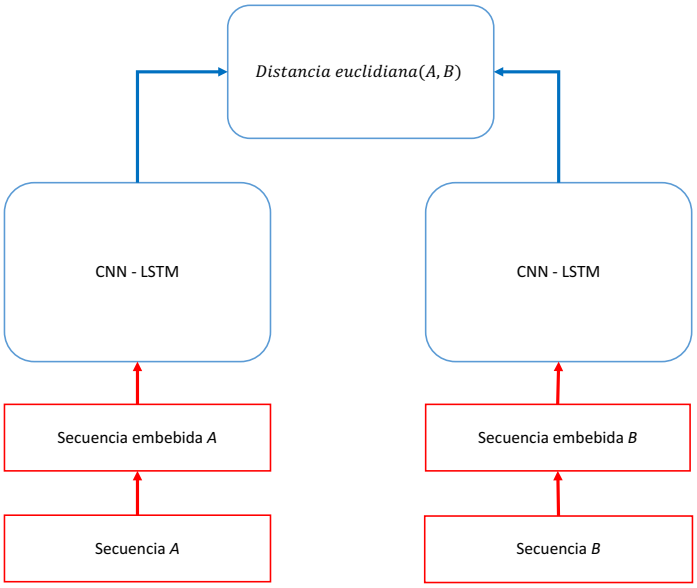

Figura 4.2: Arquitectura siamés

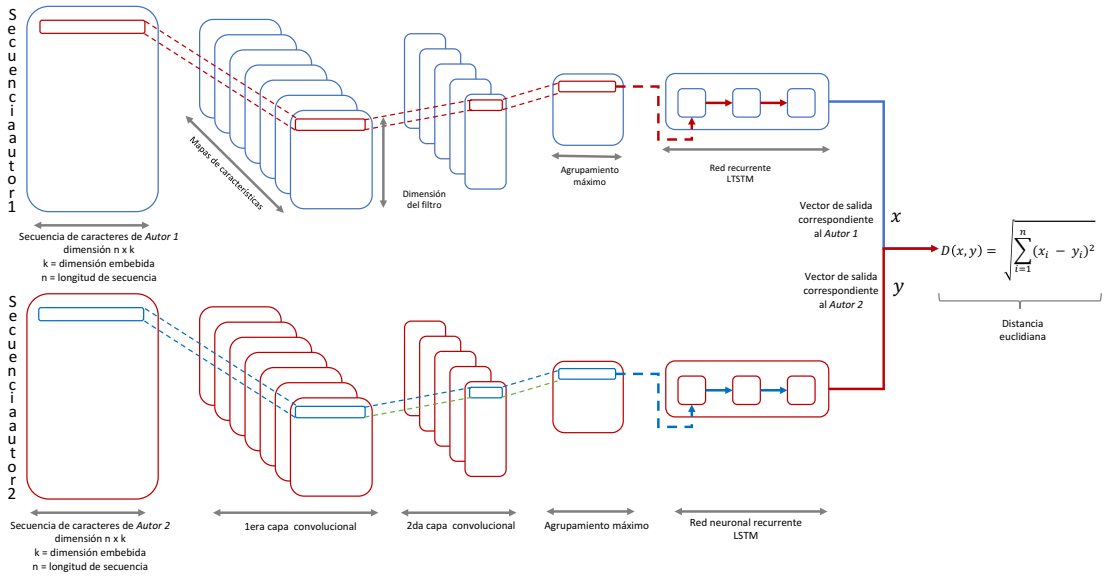

Figura 4.3: Representación del procesamiento mediante el ensamble de dos capas de redes convolucionales, una capa de agrupamiento máximo y una red recurrente de memoria de corto y largo plazo.

Capítulo 5

Experimentos y resultados

Los experimentos realizados para este trabajo de tesis, están divididos en dos partes. Una primera parte corresponde a los basados en bosques aleatorios, los cuales tuvieron como finalidad generar un modelo que funja como línea base de comparación con el modelo propuesto basado en redes neuronales. En segunda instancia tenemos los experimentos desarrollados a partir de modelos de aprendizaje profundo, los cuales abarcan diferentes arquitecturas en las redes neuronales, variaciones de los parámetros e híperparámetros. Asimismo, se realizaron variaciones en la organización de los datos de entrada, es decir, se realizaron pruebas con distintas longitudes de vectores de entrada.

5.1. Conjunto de datos

El conjunto de datos fue tomado del proyecto Gutenberg [31]. El proyecto Gutenberg provee de una basta colección de textos en amplia variedad de géneros, lenguajes y tipos de autor. La selección de textos que conforman el conjunto de datos fue realizada manualmente. Esto debido a que varios documentos que provee el proyecto Gutenberg tienen ciertas discrepancias.

Por ejemplo algunos documentos que son redactados por más de dos autores se pueden encontrar como un documento escrito individualmente por cada uno de los autores, es decir, se encuentra que el mismo documento pertenece a cada autor por separado. Otra discrepancia que se encontró es que algunos documentos están organizados en orden alfabético por nombre y otros por apellido, lo que nos lleva a encontrarnos con una obra con 2 entradas diferentes correspondientes a variaciones del nombre del autor, por ejemplo, por: *Fernando López* y por otra parte nos podemos encontrar el mismo documento escrito por *López Fernando*, lo que ocasiona que tengamos el mismo documento haciendo referencia a "dos autores distintos". Dichas discrepancias no están especificadas puntualmente en los términos y condiciones del proyecto Gutenberg. Por tal motivo el conjunto de datos se organizó manualmente, verificando que cada documento no se encontrara en alguno de los casos mencionados de discrepancias.

Los textos que fueron seleccionados para estar dentro del conjunto de datos cumplen con las siguientes características:

- Están escritos por un único autor
- Están redactados en el idioma inglés
- El género del texto es variable
- La longitud del texto es variable
- La cantidad de textos por autor es de 4 a 6

La cantidad de autores total es de 245, teniendo un total de 1320 textos. La base de datos se dividió en 70 % para datos de entrenamiento y 30 % para datos de validación, teniendo finalmente 172 autores para entrenamiento y 73 autores para validación.

5.1.1. Preprocesamiento del conjunto de datos basado en modelos de espacio vectorial

Con la finalidad de generar distintos conjuntos de datos basados en bolsas de palabras de n-gramas tanto a nivel carácter como de palabra como se describe en la tabla 5.8.

Partiendo del conjunto de datos original, se eliminaron formas que no pertenecen a la simbología de escritura haciendo uso de expresiones regulares, parte del texto crudo lo podemos observar en la tabla 5.3. Posteriormente, todas las letras fueron transformadas a su forma minúscula y todas las palabras de paro (*stop words*)[1] fueron retiradas de cada texto obteniendo como resultado secuencias como se muestran en la tabla 5.4. No se implementó ninguna técnica de tokenización ni lematización, es decir, se trabajó con la palabra tal como en el documento se encuentra. Cada documento fue seccionado por diferentes longitudes de n-gramas específicamente 2-gramas, 3-gramas y 4-gramas, tanto a nivel carácter como a nivel palabra como se muestra en las tablas 5.1 y 5.2. Posteriormente se calculó la frecuencia de termino - frecuencia inversa de documentos (*tf-idf* por sus siglas en inglés), teniendo como resultado vectores que representan a cada documento en términos de bolsas de palabras de n-gramas de distintas longitudes tanto a nivel carácter como a nivel palabra.

`shes the best person ive known in my life`	Texto original
`sh es th eb es tp er so ni ve kn ow ni nm yl if e`	2-gramas carácter
`she sth ebe stp ers oni vek now nin myl ife`	3-gramas carácter
`shes theb estp erso nive know ninm ylif e`	4-gramas carácter

Tabla 5.1: Representación de n-gramas a nivel carácter

[1] https://www.ranks.nl/stopwords

shes the best person ive known in my life	Texto original
shesthe bestperson iveknown inmy life	2-gramas palabra
shesthebest personiveknown inmylife	3-gramas palabra
shesthebestperson iveknowninmy life	4-gramas palabra

Tabla 5.2: Representación de n-gramas a nivel palabra

5.1.2. Preprocesamiento de la base de datos para modelo de redes neuronales

En primera instancia, se hizo una limpieza de cada documento, es decir, se eliminaron símbolos que no forman parte de la escritura así como caracteres especiales y números. Esto se llevó acabo haciendo uso de expresiones regulares. La figura 5.3 muestra un texto original con símbolos que no pertenecen a la escritura original. La figura 5.4 muestra una secuencia que fue procesada bajo expresiones regulares.

\x59 She \nstreamed McGregor stood, with a little\nknot of matrons\t

Tabla 5.3: Representación de parte de un texto sin tratar.

shes the best person ive known in my life, shes my everything.

Tabla 5.4: Ejemplo de texto tratado con expresiones regulares.

A partir de los textos tratados bajo expresiones regulares, se generó un diccionario de caracteres. Dicho diccionario está formado por los 33 caracteres que se muestran en la tabla 5.5. Posterior a la creación del diccionario, se generó su equivalente en caracteres embebidos [2], es decir, se realizó un mapeo de la base de datos de caracteres embebidos [27] al diccionario generado. En consecuencia, obtuvimos un diccionario de caracteres embebidos de dimensiones (33,300), siendo 33 el número de elementos en el diccionario y 300 la dimensión de los vectores embebidos. El uso de una transformación de caracteres simples a un espacio de caracteres embebidos es debido a que los caracteres representan datos categóricos, de lo

[2]https://github.com/minimaxir/char-embeddings

que no podríamos extraer la información que deseamos. La representación embebida de los caracteres nos da una representación en espacio de dicho carácter en relación a los demás caracteres que componen el texto.

Finalmente se obtuvo una base de datos con la misma cantidad de autores y documentos que la base de datos original, con la diferencia de que la nueva base de datos está representado por caracteres embebidos en lugar de caracteres simples.

[! , ", , , - , . , : , ; , ? , a , b , c , d , e , f , g , h , i , j , k, l , m , n , o , p , q , r , s , t , u , v , w , x , y , z]

Tabla 5.5: Diccionario de caracteres.

5.1.3. Generación de parejas de secuencias

Para abordar el problema de verificación en donde buscamos saber si dado un par de documentos, estos pertenecen o no al mismo autor, cada conjunto de datos, es decir, el conjunto de datos basado en bolsas de palabras de n-gramas así como el conjunto de datos representando por vectores embebidos, procesaron de tal manera que el vector de características queda representado por la unión de dos secuencias, en donde cada secuencia representa a cierto autor y que por consiguiente, el vector objetivo estaría dado por un 1 si ambas secuencias pertenecen al mismo autor y por un 0 si son de distinto autor. Entendamos al vector objetivo como y y al vector de características como x.

Para generar el vector objetivo con valores positivos, se realizó la combinación de todos los textos respectivos a un mismo autor. Es decir, se generaron pares de documentos que corresponde al mismo autor, los cuales fueron asignados con el valor de 1 en el vector objetivo.

Para ejemplificar cómo se generó el conjunto de datos a partir de secuencias, supongamos lo siguiente, el $autor_i$ tiene 3 textos, de estos fueron tomadas 5 secuencias de longitud 100 por

cada texto, lo que resulta en un total 15 muestras o secuencias para representar al $autor_i$. La generación de secuencias fue realizada para cada autor. A partir de estás secuencias, se realizó la generación de pares positivos como se ilustra en la tabla 5.6

Secuencias del $autor_i$	**Vector objetivo**
$secuencia_{i,1} + secuencia_{i,2}$	1
$secuencia_{i,1} + secuencia_{i,3}$	1
$secuencia_{i,1} + secuencia_{i,4}$	1
$secuencia_{i,2} + secuencia_{i,3}$	1
$secuencia_{i,2} + secuencia_{i,4}$	1
$secuencia_{i,3} + secuencia_{i,4}$	1

Tabla 5.6: Generación del vector objetivo a partir de la combinación en pares de las distintas secuencias obtenidas por el $autor_i$. En el cuadro se muestra que el $autor_i$ tiene 4 secuencias, las cuales son concatenadas para formar el vector objetivo.

El vector objetivo con valores negativos, es formado a partir de la combinación entre las secuencias de diferentes autores. Tal combinación es realizada de la misma manera en cómo se generó el vector objetivos de valores positivos tal como se ilustra en la tabla 5.7.

Secuencias del $autor_k$ **y** $autor_l$	**Vector objetivo**
$secuencia_{k,1} + secuencia_{l,1}$	0
$secuencia_{k,1} + secuencia_{l,2}$	0
$secuencia_{k,2} + secuencia_{l,1}$	0
$secuencia_{k,2} + secuencia_{l,1}$	0

Tabla 5.7: Generación del vector objetivo a partir de la combinación en pares de las distintas secuencias obtenidas por el $autor_k$ y el $autor_l$. En el cuadro se muestra que el $autor_k$ tiene 2 secuencias, al igual que el $autor_l$, sus secuencias son concatenadas para formar el vector objetivo con valor 0.

.

Posteriormente se realiza la unión de los vectores de características positivos y negativos, ordenándolos de manera aleatoria. De esta manera es generado el conjunto de datos total para entrenamiento tanto para el modelo de espacio vectorial como para el modelo de redes neuronales profundas. La misma dinámica se sigue para generar los datos de validación. Es importante resaltar que los datos de validación fueron aislados desde la organización en los

directorios, para que de esta manera al realizar las combinaciones, ningún elemento que está en los datos de entrenamiento se encuentre en los datos de validación.

La construcción de cada conjunto de datos tanto para el modelo de espacio vectorial como para la red neuronal, parte de la idea de que cada elemento del conjunto de datos representa a la unión de dos secuencias en donde cada secuencia representa al mismo o a distinto autor, teniendo al vector objetivo siendo 1 ó 0 dependiendo de que, el par de secuencias pertenezcan al mismo o a diferente autor, respectivamente. El total de elementos que componen a cada conjunto de datos se describe en la tabla 5.10

En la tabla 5.8 se muestran el total de conjuntos de datos generados a base de secuencias de n-gramas para el modelo de espacio vectorial. En la tabla 5.9 se muestran el total de conjuntos de dato generados a base de vectores embebidos para el modelo basado en redes neuronales.

Conjunto de datos	**n-gramas**	**Nivel**	**Longitud de secuencia**
Conjunto de datos 1	2-gramas	carácter	1000
Conjunto de datos 2	2-gramas	palabra	1000
Conjunto de datos 3	3-gramas	carácter	1000
Conjunto de datos 4	3-gramas	palabra	1000
Conjunto de datos 5	4-gramas	carácter	1000
Conjunto de datos 6	4-gramas	palabra	1000

Tabla 5.8: Conjuntos de datos generados basados en bolsas de palabras de n-gramas

Conjunto de datos	**Nivel**	**Longitud de secuencia**
Conjunto de datos 1	carácter	200
Conjunto de datos 2	carácter	400
Conjunto de datos 3	carácter	800

Tabla 5.9: Conjuntos de datos generados basados en vectores embebidos

	Datos de entrenamiento	**Datos de validación**
Número de positivos	21,686	5,422
Número de negativos	21,686	5,422
Total de elementos	43,372	10,844

Tabla 5.10: Total de elementos generados para cada conjunto de datos así como el balance de elementos tanto positivos como negativos.

5.2. Experimentos con modelos de espacio vectorial

Se realizaron experimentos con el clasificador de bosques aleatorios con vectores de 1000 elementos los cuales fueron extraídos de bolsas de palabras de n-gramas tanto a nivel carácter como a nivel palabra. En la tabla 5.11 se muestran los resultados obtenidos en los experimentos realizados. Se observa que el modelo basado en 3-gramas de caracteres arroja una exactitud ligeramente mayor que los otros modelos en donde la precisión que está capturando ligeramente más elementos positivos que negativos. La precisión del modelo de 4-gramas basado en caracteres indica que el modelo tuvo un mejor rendimiento al clasificar elementos positivos, sin embargo también se observa que captura muchos falsos positivos.

Nivel	**Exactitud**	**AUC**	**Exhaustividad**	**Precisión**	**F1-score**
2-gramas carácter	0.6743	0.6532	0.6789	0.7012	0.6898
2-gramas palabra	0.5509	0.5423	0.5282	0.5981	0.5609
3-gramas carácter	0.7188	0.7392	0.6991	0.6835	0.6912
3-gramas palabra	0.5212	0.5021	0.5364	0.5298	0.5330
4-gramas carácter	0.7079	0.7112	0.6521	0.7275	0.6877
4-gramas palabra	0.5274	0.5132	0.5326	0.5412	0.5368

Tabla 5.11: Resultados de experimentos basados en bolsas de palabras de n-gramas bajo el clasificador de bosques aleatorios.

5.3. Experimentos con ensamble de redes neuronales profundas

En esta sección detallaremos los resultados obtenidos al desarrollar el ensamble de una red convolucional y una red recurrente de memoria de corto a largo plazo. Dichos experimentos fueron realizados haciendo uso de la base de datos del proyecto Gutenberg [31]. Asimismo, se presentan los resultados de aplicar el modelo entrenado a las bases de datos proporcionadas por la competencia PAN CLEF 2013 y 2014[3] [4] con la finalidad de mostrar la generalidad del modelo. La arquitectura siamés presentada en el capítulo 4 fue implementada haciendo variaciones tanto en la longitud de las secuencias así como en la ventana de agrupamiento.

5.3.1. Variaciones en capa de agrupamiento con secuencias de 200 caracteres

Cada entrada al modelo fue dada por una secuencia de 200 caracteres embebidos por cada autor. Las variaciones se muestran a continuación.

- Capa de agrupamiento máximo con ventana de 4 elementos
- Capa de agrupamiento máximo con ventana de 6 elementos
- Capa de agrupamiento máximo con ventana de 8 elementos

En la figura 5.1 se muestra la pérdida y la exactitud para los datos de entrenamiento y de validación representados por secuencias de 200 caracteres para cada autor y 4 elementos en la capa de agrupamiento máximo. En la figura 5.2 se muestra el resultado de aplicar 6 elementos

[3] http://pan.webis.de/clef13/pan13-web/author-identification.html
[4] http://pan.webis.de/clef14/pan14-web/author-identification.html

en la capa de agrupamiento máximo y en la figura 5.3 se muestra el resultado de aplicar 8 elementos en la capa de agrupamiento máximo.

Para los experimentos con secuencias de 200 caracteres por autor y con variaciones en la capa de agrupamiento máximo con 4, 6 y 8 elementos, se observa que el rendimiento es bajo, es decir, el modelo aprende características pero no son las suficientes para poder determinar si el par de secuencias ingresadas pertenecen o no al mismo autor. Esto se debe a que la secuencia es corta, es decir, 200 caracteres parecen insuficientes para poder describir el estilo de un autor. Otra característica que se observa es que mientras más grande es la ventana de agrupamiento máximo, el modelo deja de aprender características debido a que la secuencia que entra memoria de corto y largo plazo es muy corta.

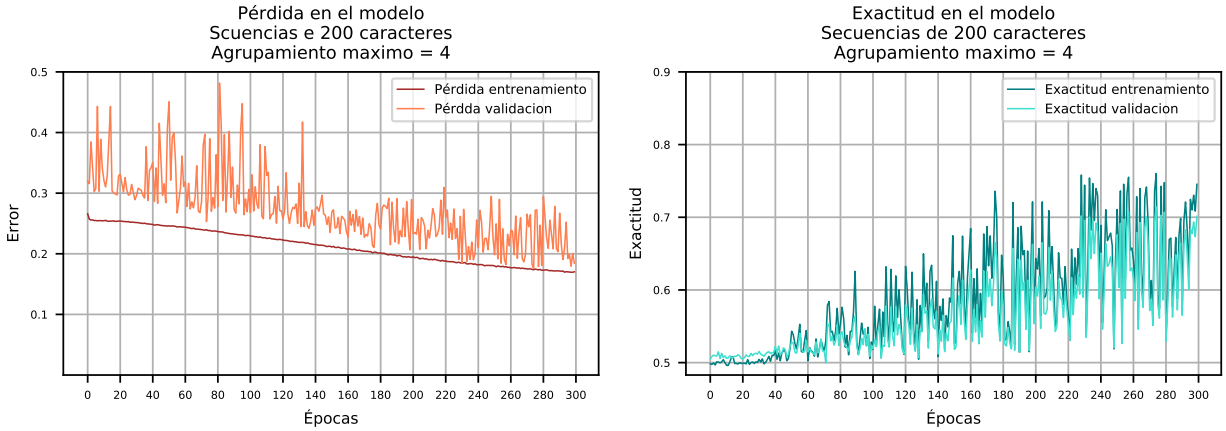

Figura 5.1: Rendimiento del modelo aplicando una capa de agrupamiento máximo con ventana de 4 elementos a una secuencia de 200 caracteres por autor.

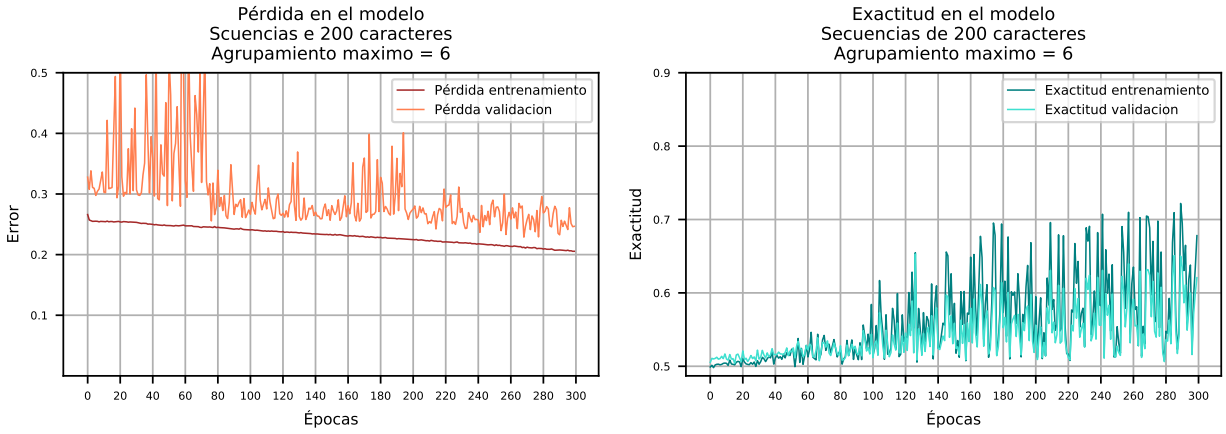

Figura 5.2: Rendimiento del modelo aplicando una capa de agrupamiento máximo con ventana de 6 elementos a una secuencia de 200 caracteres por autor.

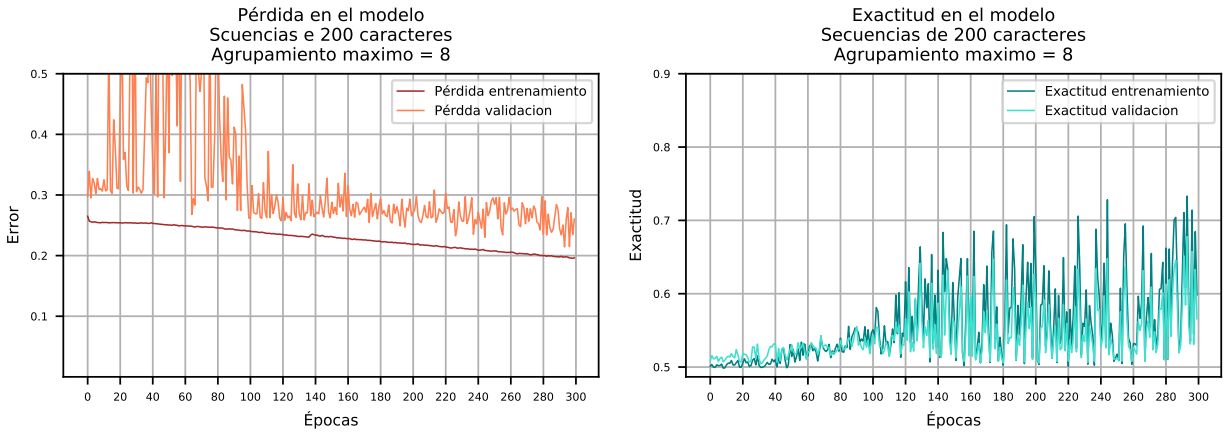

Figura 5.3: Rendimiento del modelo aplicando una capa de agrupamiento máximo con ventana de 8 elementos a una secuencia de 200 caracteres por autor.

5.3.2. Variaciones en capa de agrupamiento con secuencias de 400 caracteres

Cada entrada al modelo fue dada por una secuencia de 400 caracteres embebidos para cada autor. Las variaciones son mostradas a continuación. En la figura 5.4 se muestra la pérdida y la exactitud para los datos de entrenamiento y de validación representados por secuencias

de 400 caracteres para cada autor y 4 elementos en la capa de agrupamiento máximo. En la figura 5.5 se muestra el resultado de aplicar 6 elementos en la capa de agrupamiento máximo, asimismo en la figura 5.6 se muestra el resultado de aplicar 8 elementos en la capa de agrupamiento máximo.

Para los experimentos con secuencias de 400 caracteres por autor y con variaciones en la capa de agrupamiento máximo con 4, 6 y 8 elementos, se observa que el modelo aprende características de cada autor y es ligeramente mejor que el modelo con secuencias de 200 elemento. Sin embargo, aún es un modelo con una exactitud baja, es decir, el modelo aprende algunas características pero no son las suficientes para poder describir a un autor en base a su escritura. La secuencia de 400 caracteres no es corta, sin embargo, no son suficientes para poder aprender de manera adecuada las características que definen a cada autor.

- Capa de agrupamiento máximo con ventana de 4 elementos
- Capa de agrupamiento máximo con ventana de 6 elementos
- Capa de agrupamiento máximo con ventana de 8 elementos

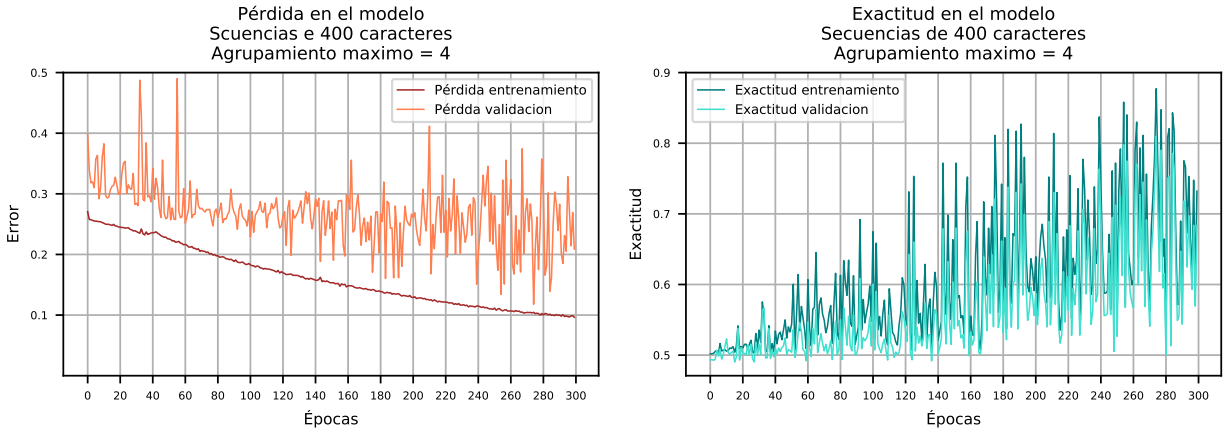

Figura 5.4: Rendimiento del modelo aplicando una capa de agrupamiento máximo con ventana de 4 elementos a una secuencia de 400 caracteres por autor.

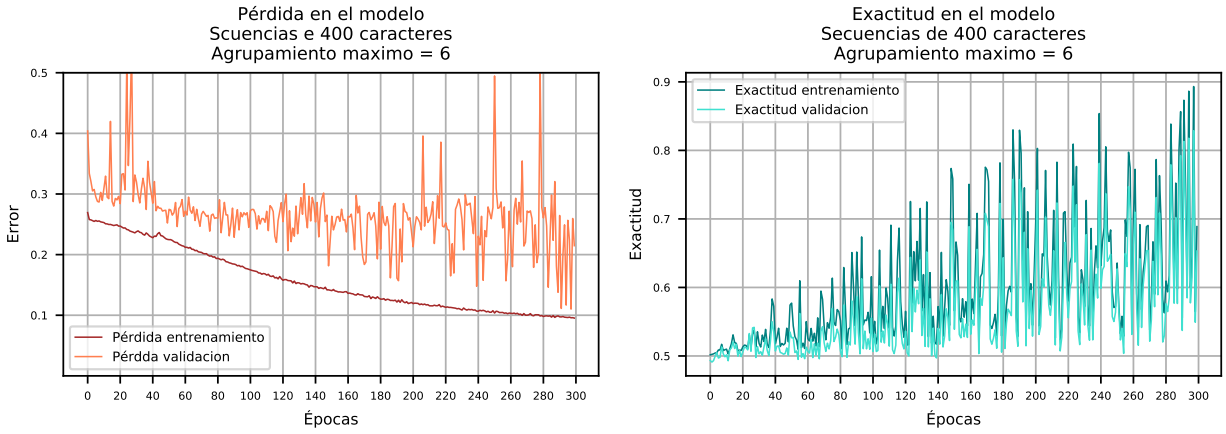

Figura 5.5: Rendimiento del modelo aplicando una capa de agrupamiento máximo con ventana de 6 elementos a una secuencia de 400 caracteres por autor.

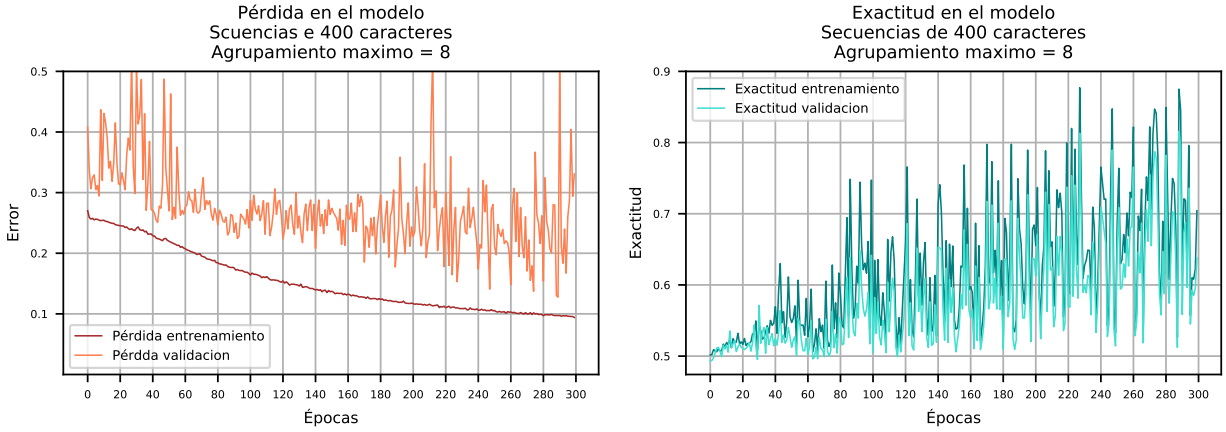

Figura 5.6: Rendimiento del modelo aplicando una capa de agrupamiento máximo con ventana de 8 elementos a una secuencia de 400 caracteres por autor.

5.3.3. Variaciones en capa de agrupamiento con secuencias de 800 caracteres

Cada entrada al modelo se da por una secuencia de 800 caracteres embebidos por cada autor.

Las variaciones se muestran a continuación.

- Capa de agrupamiento máximo con ventana de 4 elementos
- Capa de agrupamiento máximo con ventana de 6 elementos
- Capa de agrupamiento máximo con ventana de 8 elementos

En la figura 5.7 se muestra la pérdida y la exactitud para los datos de entrenamiento y de validación representados por secuencias de 800 caracteres para cada autor y 4 elementos en la capa de agrupamiento máximo. En la figura 5.8 se muestra el resultado de aplicar 6 elementos en la capa de agrupamiento máximo y en la figura 5.9 se muestra el resultado de aplicar 8 elementos en la capa de agrupamiento máximo.

El modelo con 800 caracteres fue entrenado en 100 épocas debido a que a partir de la época 50 el rendimiento se mantiene constante en las 3 variaciones realizadas. A diferencia de los modelos con secuencias de 200 y 400 caracteres para cada autor, el modelo con 800 caracteres muestra un desempeño aceptable y apegado al estado del arte. Se observa que mientras más grande es la capa de agrupamiento máximo, el rendimiento del modelo tiende a desestabilizarse.

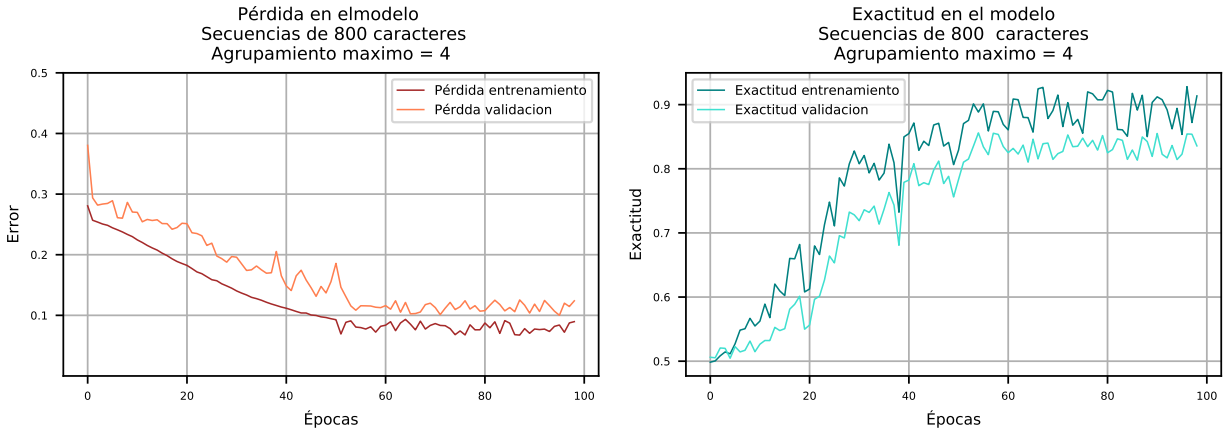

Figura 5.7: Rendimiento del modelo aplicando una capa de agrupamiento máximo con ventana de 4 elementos a una secuencia de 800 caracteres por autor.

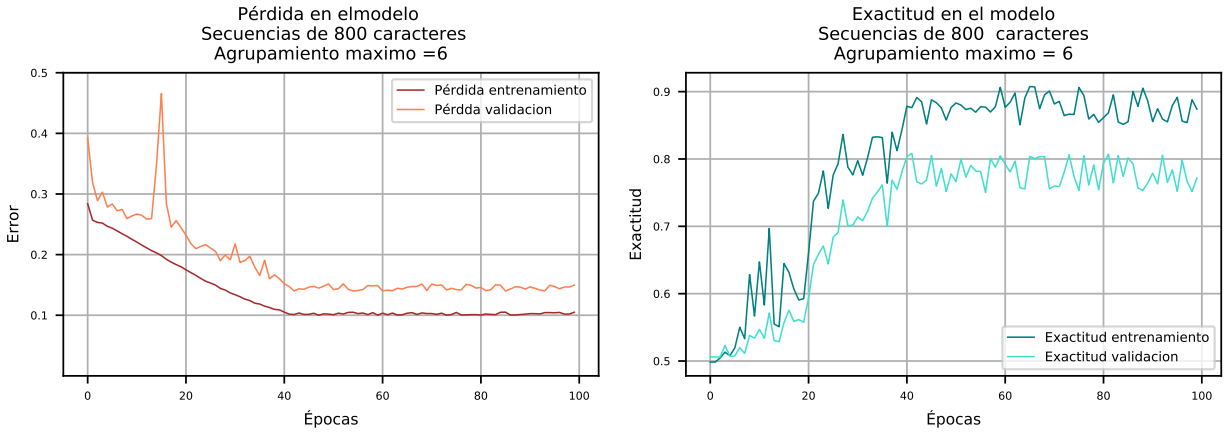

Figura 5.8: Rendimiento del modelo aplicando una capa de agrupamiento máximo con ventana de 6 elementos a una secuencia de 800 caracteres por autor.

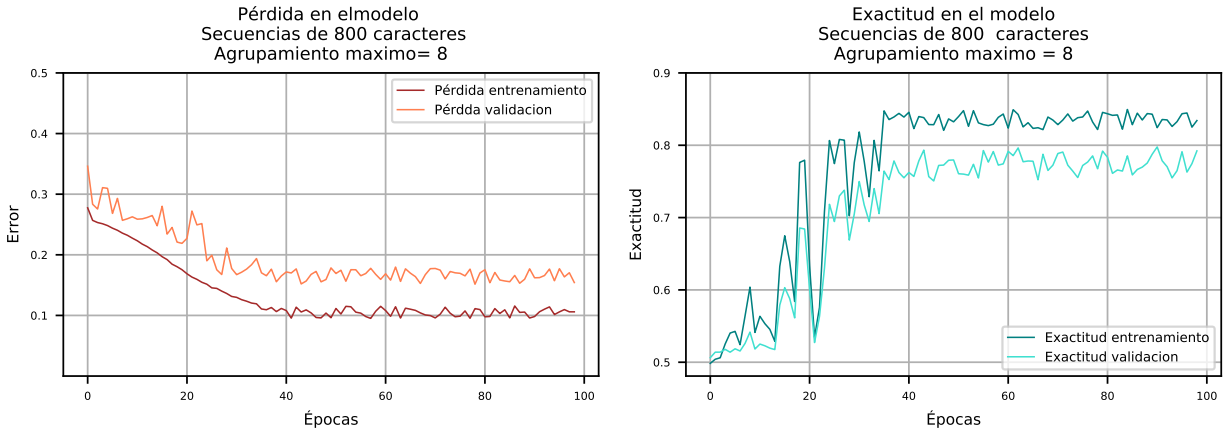

Figura 5.9: Rendimiento del modelo aplicando una capa de agrupamiento máximo con ventana de 8 elementos a una secuencia de 800 caracteres por autor.

Una de las métricas utilizadas para medir el rendimiento del modelo en la tarea de verificación de autor fue la curva ROC. Dicha curva describe de manera gráfica el comportamiento del modelo dados los verdaderos positivos con respecto a los falsos positivos.

La figura 5.10 muestra el desempeño de los modelos con una ventana de agrupamiento máximo de 4 elementos. En este caso el modelo con mejor rendimiento es el que trabaja con secuencias de 800 caracteres por autor. Se observa que el modelo con rendimiento más bajo para este caso

es el que trabajo con secuencias de 400 caracteres. En la figura 5.11 se muestra el desempeño de los modelos con una ventana de agrupamiento máximo de 6 elementos en donde el mejor desempeño se obtiene al trabajar con secuencias de 800 caracteres. El rendimiento más bajo se obtiene al trabajar con secuencias de 200 caracteres. Por último, el modelo con una venta de agrupamiento máximo de 8 elementos de la figura 5.12 muestra el mejor desempeño al trabajar con secuencias de 800 caracteres y el desempeño más bajo se obtiene al trabajar con secuencias de 200 caracteres.

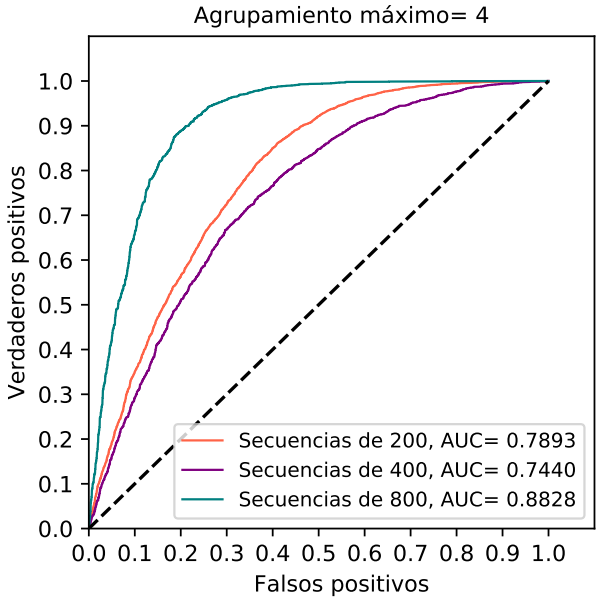

Figura 5.10: Curva ROC de las variaciones por longitud de secuencia manteniendo la ventana de agrupamiento máximo con valor de 4.

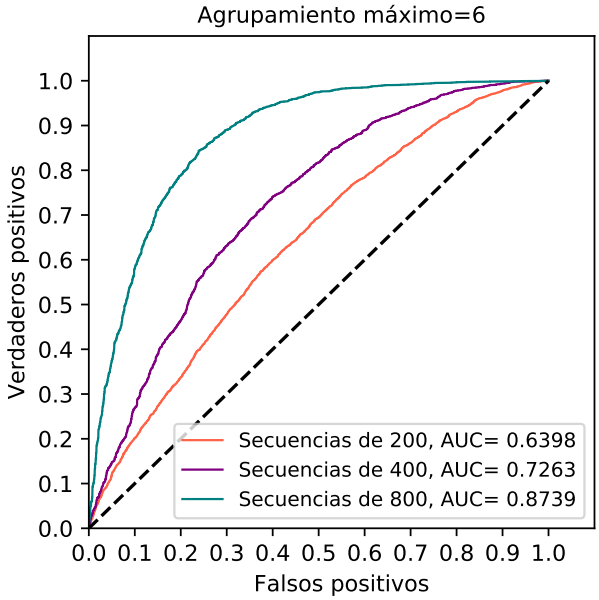

Figura 5.11: Curva ROC de las variaciones por longitud de secuencia manteniendo la ventana de agrupamiento máximo con valor de 6.

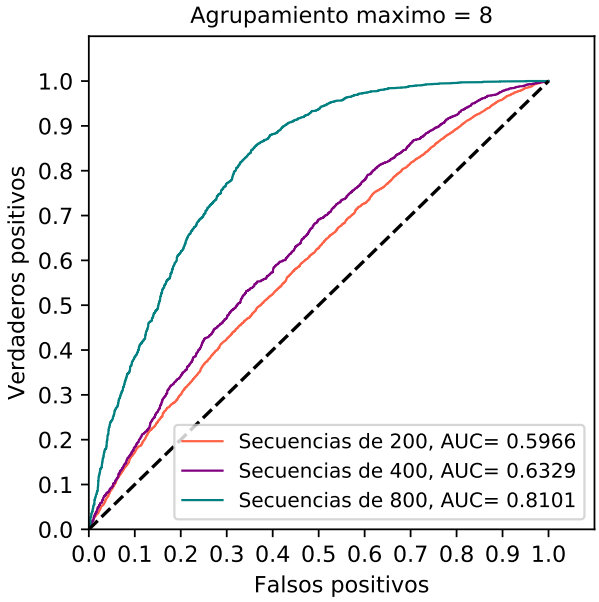

Figura 5.12: Curva ROC de las variaciones por longitud de secuencia manteniendo la ventana de agrupamiento máximo con valor de 8.

El resultado en las curvas ROC para cada una de las variaciones realizadas indica que el

mejor rendimiento se obtiene al trabajar con secuencias más largas y con una ventana de agrupamiento máximo de pocos elementos. Asimismo, el rendimiento más bajo se obtiene al trabajar con secuencias cortas. También, mientras más grande se aplica la venta de agrupamiento para cualquier longitud en la secuencia, el rendimiento decrece. En la tabla 5.12 se presentan los resultados al evaluar el modelo con datos obtenidos del proyecto Gutenberg. Se observa para los modelos evaluados con secuencias de 200 caracteres que cuando se tiene una ventana de agrupamiento máximo de 4 elementos se capturan una cantidad significativa de verdaderos positivos, sin embargo, mientras más grande se hace la venta de agrupamiento, se visualiza que la tasa de verdaderos positivos decrece mientras que los falsos positivos aumentan. El mismo efecto sucede para modelos con secuencias de 400 caracteres, teniendo éstas la particularidad de que cuando la ventana de agrupamiento máximo es de 8 elementos, prácticamente se capturan falsos positivos. El modelo con secuencias de 800 caracteres muestra tener ligera estabilidad en la captura de falsos positivos al incrementar la venta de agrupamiento máximo. [5]

Resultados obtenidos del conjunto de datos del proyecto Gutenberg						
Arquitectura	**Agrupamiento**	**Exactitud validación**	**AUC**	**Precisión**	**Exhaustividad**	**F1-score**
CNN-LSTM 200	4	0.7206	0.7893	0.7013	0.7782	0.7377
CNN-LSTM 200	6	0.5867	0.6397	0.6207	0.4670	0.5330
CNN-LSTM 200	8	0.5654	0.5966	0.5659	0.5995	0.5822
CNN-LSTM 400	4	0.6840	0.7440	0.6837	0.6690	0.6763
CNN-LSTM 400	6	0.6680	0.7263	0.6531	0.6976	0.6746
CNN-LSTM 400	8	0.5479	0.6328	0.6380	0.1934	0.2968
CNN-LSTM 800	4	0.8446	0.8828	0.8233	0.8823	0.8518
CNN-LSTM 800	6	0.7842	0.8738	0.8252	0.7278	0.7735
CNN-LSTM 800	8	0.7209	0.8101	0.7512	0.6709	0.7088

Tabla 5.12: Resumen de resultados en experimentos sobre el conjunto de datos obtenido del proyecto Gutenberg

[5]Código del modelo accesible en el repositorio: https://github.com/FernandoLpz/AuthorVerificiation

5.4. Experimentos con conjunto de datos de la competencia PAN CLEF 2013

En esta sección se presentan los resultados al evaluar el modelo con el conjunto de datos obtenida de la competencia PAN CLEF 2013 con la finalidad de probar la generalidad de dicho modelo entrenado y presentado en la sección 5.3.

5.4.1. Descripción del conjunto de datos

El conjunto de datos consta de 68 autores, cada uno con 4 o 5 documentos en donde cada uno tiene una longitud en promedio de 1400 palabras. Los documentos son fragmentos de redacciones con temas sobre ciencias de la computación y áreas relacionadas. El conjunto de datos fue preprocesado usando la misma metodología con el que se preprocesó el conjunto de datos extraído del proyecto Gutenberg . La cantidad de ejemplos posteriores al preprocesamiento y con los que se probó el modelo se presentan en la tabla 5.13

Datos para prueba	
Número de positivos	4,759
Número de negativos	4,759
Total de elementos	9,518

Tabla 5.13: Estadísticas de los ejemplos recopilados del conjunto de datos obtenido de PAN CLEF 2013

5.4.2. Resultados de experimentos

Se realizaron experimentos a partir de los modelos ya entrenados y mostrados en la sección 5.3. Esto con la finalidad de probar la generalidad del modelo en una base de datos diferente a con la que se entrenó. El modelo bajo el cual se desarrollaron los experimentos fue el de

mejor desempeño al validar con la base de datos del proyecto Gutenberg, es decir, el modelo entrenado con secuencias de 800 caracteres. De la misma manera, se realizaron variaciones en la ventana de agrupamiento máximo.

En la figura 5.13 se muestra la curva ROC del desempeño del modelo. Se observa que el mejor desempeño se obtiene al trabajar con una ventana de agrupamiento máximo de 4 elementos mientras que el desempeño más bajo se obtiene al trabajar con una venta de 8 elementos lo que indica que con una secuencia de 800 caracteres se puede aprender el estilo de un autor sin embargo, mientras más grande es la ventana de agrupamiento máximo, el desempeño tiende a decrecer. En la tabla 5.14 se presentan los resultados al probar con datos obtenidos de la competencia PAN CLEF 2013. Se observa para los modelos evaluados con secuencias de 200 caracteres que cuando se tiene una ventana de agrupamiento máximo de 4 elementos se capturan una cantidad significativa de verdaderos positivos, sin embargo, se observa que mientras la ventana de agrupamiento máximo aumenta, al exhaustividad decrece ligeramente. Esto indica que la tasa de falsos positivos incremento ligeramente sin embargo este efecto no afecta de manera significativa el rendimiento del modelo.

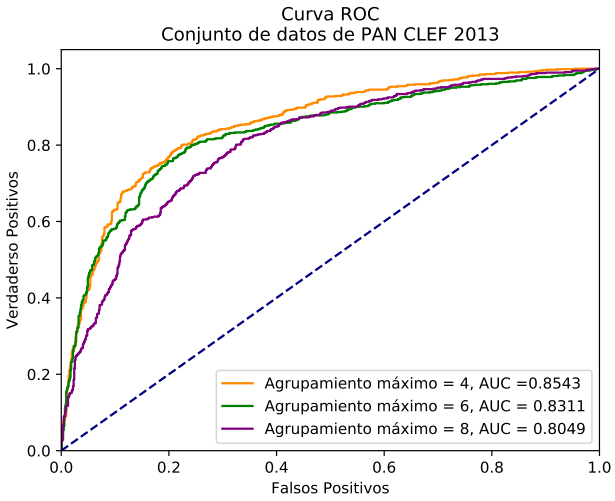

Figura 5.13: Curva ROC con variaciones en la capa de agrupamiento máximo.

Resultados obtenidos del conjunto de datos de PAN CLEF 2013						
Arquitectura	**Agrupamiento**	**Exactitud**	**AUC**	**Precisión**	**Exhaustividad**	**F1-score**
CNN-LSTM 800	4	0.7850	0.8543	0.7789	0.7630	0.7709
CNN-LSTM 800	6	0.7548	0.8311	0.8003	0.6433	0.7132
CNN-LSTM 800	8	0.7288	0.8048	0.7420	0.6560	0.6964

Tabla 5.14: Resumen de resultados en experimentos sobre el conjunto de datos obtenido de la competencia PAN CLEF 2013.

5.5. Experimentos con conjunto de datos de la competencia PAN CLEF 2014

En esta sección se presentan los resultados al evaluar el modelo con la base de datos obtenida de la competencia PAN CLEF 2014 con la finalidad de probar la generalidad de dicho modelo entrenado y presentado en la sección 5.3.

5.5.1. Descripción del conjunto de datos

Del conjunto de datos original se eligieron 200 autores, cada uno con 4 o 5 documentos en donde cada uno tiene una longitud en promedio de 1100 palabras. Cada documento es una redacción del género novela. El conjunto de datos fue preprocesada usando la misma metodología con la que se preprocesó el conjunto de datos extraído del proyecto Gutenberg . Los elementos posteriores al preprocesamiento y con los que se probó el modelo se presentan en la tabla 5.15

Datos para prueba	
Número de positivos	14,244
Número de negativos	14,244
Total de elementos	28,488

Tabla 5.15: Estadísticas de los ejemplos recopilados del conjunto de datos obtenido de PAN CLEF 2014

5.5.2. Resultados de experimentos

En la figura 5.14 se muestra la curva ROC del desempeño del modelo. Cada experimento se realizó con secuencias de 800 caracteres por autor y con variaciones en la capa de agrupamiento máximo. Se observa que el mejor desempeño se obtiene al trabajar con una ventana de agrupamiento máximo de 4 elementos mientras que el desempeño más bajo se obtiene al trabajar con una venta de 8 elementos lo que indica que con una secuencia de 800 caracteres se puede aprender el estilo de un autor sin embargo, mientras más grande es la ventana de agrupamiento máximo, el desempeño tiende a decrecer. En la tabla 5.16 se presentan los resultados al probar el modelo con datos obtenidos de la competencia PAN CLEF 2014. La precisión del modelo para las tres variaciones de ventana de agrupamiento máximo nos indica que se capturan una cantidad significativa de elementos positivos. Sin embargo, cuando la ventana de agrupamiento máximo aumenta, incrementan los falsos positivos ligeramente no afectando de manera puntual el rendimiento del modelo.

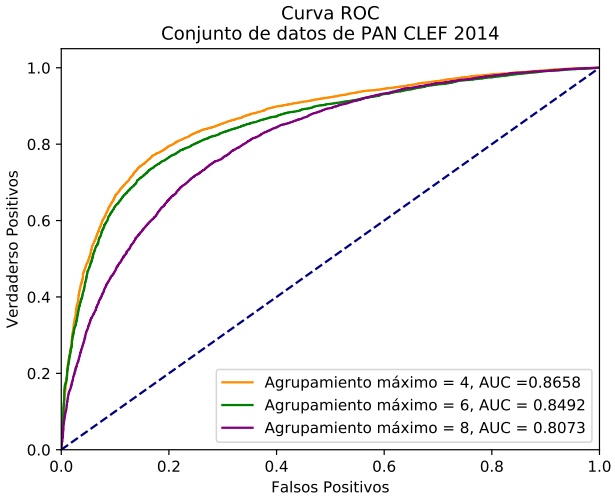

Figura 5.14: Curva ROC con variaciones en la capa de agrupamiento máximo.

Resultados obtenidos del conjunto de datos de PAN CLEF 2014						
Arquitectura	**Agrupamiento**	**Exactitud**	**AUC**	**Precisión**	**Exhaustividad**	**F1-score**
CNN-LSTM 800	4	0.7996	0.8658	0.8129	0.7746	0.7933
CNN-LSTM 800	6	0.7788	0.8492	0.8364	0.6893	0.7558
CNN-LSTM 800	8	0.7266	0.8072	0.7655	0.6476	0.7017

Tabla 5.16: Resumen de resultados en experimentos sobre el conjunto de datos obtenido de la competencia PAN CLEF 2014.

Capítulo 6

Conclusiones

En este trabajo de tesis se presenta un método general para el problema de verificación de autor en base a su estilo de escritura el cual está basado en una arquitectura siamés conformada por un ensamble de redes neuronales.

Se realizó un modelo de comparación base el cual parte del uso de modelos de espacio vectorial. El modelo fue creado a partir de bolsas de palabras de distintas variaciones de n-gramas tanto a nivel carácter como a nivel palabra y que posteriormente se clasificaron haciendo uso del método de bosques aleatorios. Los resultados obtenidos en los experimentos realizados sobre el modelo base muestran tener un desempeño bajo en general. Particularmente se observa que al trabajar con n-gramas de caracteres se puede llegar a tener un desempeño ligeramente mejor que al trabajar con n-gramas de palabras.

El modelo propuesto se basa en una arquitectura siamés compuesta por un ensamble de redes neuronales convolucionales y LSTM. La clasificación es realizada usando la distancia ecuclidiana. El modelo se entrenó haciendo uso de un conjunto de datos obtenido del proyecto Gutenberg. Se tomaron distintas longitudes de secuencias de caracteres lo cuales fueron

mapeados a caracteres embebidos. Se observó que mientras más larga es la secuencia de caracteres con la que se entrena el modelo se capturan mas verdaderos positivos y menos falsos positivos. Asimismo se observó que mientras más grande es la ventana de agrupamiento máximo, el rendimiento del modelo decrece ligeramente. Los modelos entrenado fueron probados con conjuntos de datos externos con la finalidad de corroborar la generalidad, estos fueron conjuntos de datos obtenidos de la competencia PAN CLEF 2013 y 2014. Los experimentos realizados con estos dos últimos conjuntos de datos corroboran la generalidad del modelo.

El modelo basado en bolsas de palabras de n-gramas en comparación con el modelo basado en redes neuronales muestra resultados por debajo del estado del arte. El modelo que se propone basado en una arquitectura siamés arroja resultados apegados al estado del arte. Verificar un par de documentos en base a la escritura de cada autor, requiere de una abstracción profunda de la evidencia que se tiene. Un modelo de espacio vectorial se ve limitado por varias razones, una de las cuales es que el conteo de frecuencias de términos a nivel carácter o palabra no es suficiente debido a que la evidencia es analizada de forma categórica y no de forma granular. Por otra parte, el modelo basado en ensambles de redes neuronales se retroalimenta en cada evidencia que se le entrega al modelo.

Dadas las características de la arquitectura siamés que se propone en este trabajo de tesis, se concluye que el modelo tiene un mejor rendimiento cuando las secuencias de caracteres que representan a cada autor son de mayor longitud, asimismo, mientras mas grande es la ventana de agrupamiento máximo, el rendimiento tiende a decrecer no importando el tamaño de la secuencia de caracteres. En otras palabras, el modelo tiene la capacidad de verificar con un alto nivel de efectividad a un par de autores en base a su estilo de escritura siempre y cuando la muestra que toma el modelo sea lo suficientemente grande para poder aprender las características de un autor dado.

6.1. Trabajo futuro

Este trabajo de tesis fue desarrollado a partir de un conjunto de datos obtenido del proyecto Gutenberg en donde el conjunto de datos es limitado. Cada autor tenía a lo más 6 documentos y en donde cada documento era artículo o novela. Una de los puntos que se pueden abordar en un futuro es extender el conjunto de datos. Se sugiere obtener un conjunto de datos más extenso en donde por cada autor se tenga más de 5 documentos además de que el género de cada documento sea variable. Se sugiere también que cada documento sea de longitud variable.

La arquitectura siamés que se propone en este trabajo de tesis está compuesta por un ensamble de 2 redes convolucionales y una LSTM. Se sugiere probar con arquitecturas más profundas o con variaciones en los hiperparámetros.

El modelo que se propone trabaja con secuencias de longitud máxima de 800 caracteres embebidos. Se sugiere entrenar un modelo con secuencias de mayor longitud, la tendencia de los resultados arrojaban que mientras más larga era la secuencia, mejor era el desempeño. Asimismo se sugiere entrenar un modelo con secuencias de caracteres, de palabras y una combinación de ambos.

Bibliografía

[1] Apoorv Agarwal, Boyi Xie, Ilia Vovsha, Owen Rambow, and Rebecca Passonneau. Sentiment analysis of twitter data. In *Proceedings of the workshop on languages in social media*, pages 30–38. Association for Computational Linguistics, 2011.

[2] Douglas Bagnall. Author identification using multi-headed recurrent neural networks. *arXiv preprint arXiv:1506.04891*, 2015.

[3] Jason Brownlee. *Deep learning with Python*. Machine Learning Mastery, 2016.

[4] Omar Canales, Vinnie Monaco, Thomas Murphy, Edyta Zych, John Stewart, Charles Tappert Alex Castro, Ola Sotoye, Linda Torres, and Greg Truley. A stylometry system for authenticating students taking online tests. *P. of Student-Faculty Research Day, Ed., CSIS. Pace University*, 2011.

[5] Daniel Castro, Yaritza Adame, María Peláez Brioso, and Rafael Muñoz. Authorship verification, combining linguistic features and different similarity functions. In *CLEF (Working Notes)*, 2015.

[6] Sumit Chopra, Raia Hadsell, and Yann LeCun. Learning a similarity metric discriminatively, with application to face verification. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 539–546. IEEE, 2005.

[7] Walter Daelemans. Explanation in computational stylometry. In *International Conference on Intelligent Text Processing and Computational Linguistics*, pages 451–462. Springer, 2013.

[8] TensorFlow Developers. Motivation: Why learn word embeddings?, 2017.

[9] Robert Dorfman. A formula for the gini coefficient. *The Review of Economics and Statistics*, pages 146–149, 1979.

[10] Alex Graves. Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*, 2013.

[11] Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. Speech recognition with deep recurrent neural networks. In *Acoustics, speech and signal processing (icassp), 2013 ieee international conference on*, pages 6645–6649. IEEE, 2013.

[12] Alex Graves and Jürgen Schmidhuber. Offline handwriting recognition with multidimensional recurrent neural networks. In *Advances in neural information processing systems*, pages 545–552, 2009.

[13] Hua He, Kevin Gimpel, and Jimmy J Lin. Multi-perspective sentence similarity modeling with convolutional neural networks. In *EMNLP*, pages 1576–1586, 2015.

[14] Geoffrey Hinton. rmsprop: Divide the gradient by a running average of its recent magnitude.

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

[16] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning*, pages 448–456, 2015.

[17] Farkhund Iqbal, Rachid Hadjidj, Benjamin CM Fung, and Mourad Debbabi. A novel approach of mining write-prints for authorship attribution in e-mail forensics. *digital investigation*, 5:S42–S51, 2008.

[18] Mahmoud Khonji and Youssef Iraqi. A slightly-modified gi-based author-verifier with lots of features (asgalf). *CLEF (Working Notes)*, 1180:977–983, 2014.

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

[20] Moshe Koppel and Yaron 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.

[21] Thomas Kurbiel and Shahrzad Khaleghian. Training of deep neural networks based on distance measures using rmsprop. *arXiv preprint arXiv:1708.01911*, 2017.

[22] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

[23] Paola Ledesma, Gibran Fuentes, Gabriela Jasso, Angel Toledo, and Ivan Meza. Distance learning for author verification. In *Proceedings of the conference pacific association for computational linguistics, PACLING*, volume 3, pages 255–264, 2003.

[24] Promita Maitra, Souvick Ghosh, and Dipankar Das. Authorship verification-an approach based on random forest. *arXiv preprint arXiv:1607.08885*, 2016.

[25] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.

[26] Erwan Moreau, Arun Jayapal, Gerard Lynch, and Carl Vogel. Author verification: basic stacked generalization applied to predictions from a set of heterogeneous learners. *Working Notes Papers of the CLEF*, 2015.

[27] Colling Moris. Dissecting google's billion word language model part 1: Character embeddings, 2016.

[28] Frederick Mosteller and David Wallace. Inference and disputed authorship: The federalist. 1964.

[29] Paul Neculoiu, Maarten Versteegh, Mihai Rotaru, and Textkernel BV Amsterdam. Learning text similarity with siamese recurrent networks. *ACL 2016*, page 148, 2016.

[30] Christopher Olah. Understanding lstm networks. *GITHUB blog, posted on August*, 27:2015, 2015.

[31] Project Gutenberg Organization. Project gutenberg, 2017.

[32] Timo Petmanson. Authorship identification using correlations of frequent features.

[33] Juan-Pablo Posadas-Durán, Ilia Markov, Helena Gómez-Adorno, Grigori Sidorov, Ildar Batyrshin, Alexander Gelbukh, and Obdulia Pichardo-Lagunas. Syntactic n-grams as features for the author profiling task. In *Proceedings of the CLEF*, 2015.

[34] Dylan Rhodes. Author attribution with cnns. *Avaiable online: https://www. semanticscholar. org/paper/Author-Attribution-with-Cnn-s-Rhodes/0a904f9d6b47dfc574f681f4d3b41bd840871b6f/pdf (accessed on 22 August 2016)*, 2015.

[35] Kamil Safin and Rita Kuznetsova. Style breach detection with neural sentence embeddings. *Working Notes Papers of the CLEF*, 2017.

[36] Yunita Sari. Gender identification for adversarial writing. *ESSLLI 2015 Student Session*, page 204.

[37] Shachar Seidman. Authorship verification using the impostors method. In *CLEF 2013 Evaluation Labs and Workshop-Online Working Notes*, 2013.

[38] Gregory Shalhoub, Robin Simon, Ramesh Iyer, Jayendra Tailor, and Sandra Westcott. Stylometry system–use cases and feasibility study. *Forensic Linguistics*, 1(8), 2010.

[39] Prasha Shrestha, Sebastian Sierra, Fabio A González, Paolo Rosso, Manuel Montes-y Gómez, and Thamar Solorio. Convolutional neural networks for authorship attribution of short texts. *EACL 2017*, page 669, 2017.

[40] Julián Solórzano, Víctor Mijangos, Alejandro Pimentel, Fernanda López-Escobedo, Azucena Montes, and Gerardo Sierra. Authorship verification by combining svms with kernels optimized for different feature categories. *Working Notes Papers of the CLEF*, 2015.

[41] Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *Journal of machine learning research*, 15(1):1929–1958, 2014.

[42] Tijmen Tieleman and Geoffrey Hinton. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*, 4(2):26–31, 2012.

[43] Cor J Veenman and Zhenshi Li. Authorship verification with compression features. In *CLEF (Working Notes)*, 2013.