UNIVERSIDAD DE CHILE
FACULTAD DE CIENCIAS FÍSICAS Y MATEMÁTICAS
DEPARTAMENTO DE INGENIERÍA ELÉCTRICA

BUSCADOR FONÉTICO EN BASE DE REDES NEURONALES

MEMORIA PARA OPTAR AL TÍTULO DE INGENIERO CIVIL ELÉCTRICO

MAURICIO ANGELO ANTONELLI ORTIZ

PROFESOR GUÍA:
PROF. CLAUDIO PÉREZ FLORES

MIEMBROS DE LA COMISIÓN:
PROF. FRANCISCO RIVERA SERRANO
PROF. EDUARDO VERA SOBRINO

SANTIAGO DE CHILE
2022

Resumen

RESUMEN DE LA MEMORIA PARA OPTAR
AL TÍTULO DE: Ingeniero Civil Eléctrico
POR: Mauricio Angelo Antonelli Ortiz
FECHA: 2022
PROFESOR GUÍA: Prof. Claudio Pérez Flores

BUSCADOR FONÉTICO EN BASE DE REDES NEURONALES

El Instituto Nacional de Propiedad Industrial (INAPI) es la entidad que registra las marcas en Chile y contiene una base de datos de más de 500.000 marcas. Cada vez que registra una nueva marca debe realizar una búsqueda en sus bases de datos para verificar que esta nueva marca no sea igual o similar a una marca ya registrada, por lo que es importante desarrollar métodos de búsqueda fonética para marcas en español e inglés más rápidos y eficientes que los métodos que ya existen.

El presente trabajo muestra el desarrollo de un método de búsqueda de palabras a base de Redes Neuronales que permite encontrar marcas en una base de datos parecidas fonéticamente a una marca buscada, cuya velocidad y precisión se comparó con métodos tradicionales. También, se trabajó en un objetivo secundario de extender una base de datos de marcas fonéticamente similares que sirvió para entrenar y probar los métodos.

Para lograr estos objetivos, primero se estudió la literatura necesaria para entender mejor el problema, las soluciones actuales al problema y las posibles soluciones a base de redes neuronales que se intentaron adaptar para llevar a cabo los objetivos. Luego, se extendió la base de datos, tanto agregando pares de marcas manualmente como buscado en repositorios públicos, y se implementaron los códigos de los algoritmos. Entonces, se midió el ranking promedio de estos algoritmos para comparar sus precisiones y, también, se estimó el tiempo de procesamiento promedio por pares de marcas de los buscadores fonéticos para comparar sus velocidades.

Como resultado se obtuvo una base de datos mucho más grande que la original, la que se utilizó para entrenar las redes neuronales y comparar la precisión de los algoritmos implementados. También, se logró implementar un algoritmo que obtuvo mejores resultados de precisión que los métodos tradicionales, pero con una velocidad más baja que la esperada. Pese a esto, se encontraron otras configuraciones de este algoritmo que lograron una precisión un poco más baja que la original, aunque aún más alta que la de los métodos tradicionales, a cambio de velocidades un poco más cercanas a las de los métodos tradicionales.

Agradecimientos

Al Profesor Claudio Pérez, quién me abrió las puertas al Laboratorio de Procesamiento Digital de Imágenes. También, me permitió realizar práctica y memoria cuando estaba en una situación complicada debido a la pandemia.

A Juan Pablo Pérez, quien me ayudó durante la práctica de verano y en el inicio de mi trabajo de memoria, complementando las indicaciones del profesor Claudio Pérez.

A Francisco Rivera, por su buena disposición y su retroalimentación ante mis dudas durante el proceso de mi memoria.

A Eduardo Vera, por su confianza depositada en mí y que me permitió desarrollar un trabajo externo.

A mi familia, que siempre estuvo de mi lado y me apoyó en los momentos más difíciles durante mi carrera.

Tabla de Contenido

Índice de Tablas

Índice de Ilustraciones

Capítulo 1

Introducción

1.1. Identificación del Problema

Las marcas son una parte importante de la Propiedad Intelectual que contienen imágenes y/o nombres para distinguir productos o servicios. El Instituto Nacional de Propiedad Industrial (INAPI) es el instituto encargado de registrar las marcas en Chile y tiene una base de datos digital de más de 500.000 de marcas.

Cada vez que se registra una marca se debe buscar en las bases de datos si la marca ya existe o si existe una marca similar, para así evitar posibles confusiones en los clientes; pero, debido a que la base de datos es grande y va creciendo cada vez más, es importante desarrollar un método que permita buscar las marcas dentro de estas bases de datos en forma eficiente y precisa.

Aunque existen algoritmos fonéticos tradicionales para resolver el problema anterior, tales como *Soundex* o *Metaphone*, estos tienen ciertas debilidades, tales como, fallar cuando el par de palabras tiene una pequeña diferencia pero esta es suficiente para alterar, substancialmente, su estructura fonética [10] [11]. Además, son algoritmos que están diseñados para funcionar, principalmente, con nombres propios en inglés.

Pese a que se han creado variaciones de los algoritmos fonéticos tradicionales diseñados para funcionar en español, éstos de todos modos sufren problemas, donde las variantes de *Soundex* tienen alta exactitud pero baja precisión, debido a su alta dependencia a las letras iniciales y las variantes de *Metaphone* tienen mayor precisión pero presentan menor exactitud [12].

Por lo anterior, es que se considera estudiar el uso de redes neuronales, las cuales tienen una aplicación cada vez más masivo para resolver problemas en el área de procesamiento de textos. Sin embargo, su uso para resolver el problema de similitud de textos, desde el punto de vista fonético, no está tan desarrollado; existiendo menos literatura aún de redes neuronales utilizadas para el caso de similitud fonética en español.

1.2. Objetivo General

Desarrollar un método en base de Redes Neuronales de búsqueda de palabras que permita seleccionar marcas en una base de datos, parecidas fonéticamente a la marca buscada.

1.3. Objetivos Específicos

Los Objetivos Específicos son los siguientes:

- Extensión de una Base de Datos de Marcas fonéticamente similares en español y en inglés, con el fin de utilizarlas para entrenar y probar las redes neuronales implementadas.
- El método implementado debe presentar un mejor desempeño en precisión que los métodos tradicionales en la Base de Datos extendida, mencionada anteriormente.
- El método implementado debe tener un tiempo de respuesta de alrededor de 30 segundos para el caso de una base de datos de 300.000 marcas obtenidas de INAPI, siendo esta la velocidad estimada de un método tradicional.

Para lograr lo anterior, la presente memoria incluye un capítulo de Marco Teórico, donde se encuentra la literatura necesaria para entender tanto el problema que se debe resolver como conceptos claves para entender los distintos algoritmos fonéticos y métodos de aprendizaje de máquinas. Luego, en el capítulo de Estado de Arte se detallan los algoritmos encontrados en la literatura que podrían ayudar a resolver el problema planteado. En el capitulo de Metodología se explican los métodos utilizados para extender las Bases de Datos e implementar los distintos algoritmos con sus diferentes configuraciones. Entonces, en el capítulo de Resultados se muestran los resultados de aplicar las distintas metodologías, los que son analizados en el capitulo de Discusión de Resultados, concluyendo con los objetivos logrados y la contribución de la memoria durante el capítulo de Conclusión.

Capítulo 2

Marco Teórico

2.1. Criterios de Registro de Marcas

INAPI, la organización encargada de los registros de marcas comerciales dentro de Chile, presenta un documento público [1] con las directrices que detallan, tanto las prácticas internas realizadas por el instituto para cumplir las leyes de propiedad intelectual, como los reglamentos que delimitan sus funciones de registro de marcas comerciales y de signos distintivos.

En ese documento se explicitan los criterios utilizados para rechazar o aceptar el registro de nuevas marcas por si presentan una alta similitud a una marca ya registrada, tanto por su denominación (nombre distintivo de la marca) o por su etiqueta (imágenes o figuras). La denominación debe estar en el alfabeto latino, ya que no se aceptan caracteres de otros alfabetos, los que deben ser traducidos o transliterados según sea necesario [1].

Cabe destacar que se admite el registro de nuevas marcas similares a las ya existentes con tal que que no haya un traslape entre los productos o servicios ofrecidos por la nueva marca con respecto a la marca ya existente, por lo que se espera que no cause confusión en los clientes al momento de elegir una marca [1].

Dentro de los criterios para rechazar la denominación de nuevas marcas se comprenden varios factores de índole fonética que pueden producir que la marca a registrar tenga una pronunciación fonética muy similar a una ya marca ya registrada. Ejemplos de lo anterior se presenta a continuación [1]:

- Números expresados en palabras: expresar números en palabras no cambia la similitud de la denominación de la marca. Por ejemplo, la solicitud de ”MISS 60” fue rechazada ya que ya existía ”MISS SIXTY”, al igual que ”FIESTA DE LOS OCHENTA” fue rechazada por su similitud a ”LA FIESTA DE LOS 80”.
- Signos que contienen otras marcas: se consideran como semejantes las marcas que contienen de forma parcial o total una denominación de marca ya existente en su nombre. Por ejemplo, ”FUMEX-AGROFOL” no fue aceptado ya que ya existía ”AGRO FOL” ni

tampoco ”LARA ACCESORIES” fue aceptado porque ya existía ”LARA BOUTIQUE”.

- Diferencias de una o dos letras: las marcas que se diferencian sólo en una o dos letras suelen tener una alta similitud fonética y gráfica. Por ejemplo, se rechazó la marca ”BULGARI” por ser demasiado similar a la marca registrada ”BVLGARI”.
- Sonidos fonéticamente semejantes: se debe considerar que en español hay varias letras que, según el contexto, suenan igual, como la letra ”C” con la letra ”K”, ”CH” con ”SH”, etc. (ver tabla 2.1). Por ejemplo, se rechazó la aplicación de ”LA KOSA” ya que existía la marca ”LA COSA”.
- Incorporación o sustracción de elementos que no modifican su fonética: existen varios elementos que no modifican la fonética de una palabra, tales como incluir la letra H, la cual es muda en muchos contextos en el idioma español. Por ejemplo, ”DEKORA DEKKRA” se rechazó ante la marca ”D´KORA” ya que la sustitución de la letra ”E” por el apóstrofe no es suficiente para darle individualidad; y ”TRUCO” fue rechazado por su similitud a la marca registrada ”TRUCCO”.
- Marcas en idioma extranjero con semejanzas fonéticas: se debe considerar los casos en el que dos marcas en distintos idiomas tienen similitud fonética entre ellos por su pronunciación. Por ejemplo, se rechazó la marca solicitada ”TAISON” por ser similar a la marca registrada ”TYSON” por su pronunciación en inglés; y también se rechazó la marca ”COOLMATE PLUS” por ser su pronunciación en inglés similar a ”COLGATE”.

Además, en las directivas se esclarece la igualdad fonética de ciertas consonantes en algunos casos en el idioma español, la cual se puede ver resumido en la tabla 2.1.

Tabla 2.1: Tabla de ejemplo de letras que, en ciertos contextos, tienen sonidos equivalentes en español de Chile [1].

Letras con sonidos equivalentes en Chile	Letras con sonidos equivalentes en Chile
C	Z
Y	LL
C	K
S	Z
W	GÜ
I	Y
CH	SH

2.2. Algoritmos Fonéticos Tradicionales

2.2.1. *Soundex*

Soundex es un algoritmo creado para codificar fonéticamente nombres propios en inglés [10]. Tiene varias variantes, los que incluyen mejoras al algoritmo y extensiones del algoritmo

para considerar nombres en distintos idiomas. En general, consisten en sustituir consonantes de un nombre por números para obtener códigos que se utilizan para indexar los nombres. Por ejemplo, una variante popular y simple de implementar es el *Soundex* americano, el cual ha sido utilizado por el gobierno de Estados Unidos para analizar los resultados de censos de años anteriores en ese país[2] y consiste en seguir los siguientes pasos para codificar nombres en inglés:

1. Mantener la primera letra del nombre como el primer carácter del código.
2. Eliminar todas las vocales y las consonantes 'Y', 'W' y 'H'.
3. Reemplazar las letras restantes del nombre por los valores de código que aparecen en la tabla 2.2.
4. Si existen dos o más letras consecutivas (incluyendo la primera letra) que tienen la misma codificación, se deben tratar como una misma y dejar solo el primer carácter del código.
5. El código resultante debe tener 4 caracteres, por lo que se debe rellenar con ceros si faltan caracteres, o se deben eliminar los caracteres adicionales si tiene más que 4.

Con los pasos anteriores se logra obtener un código de cuatro caracteres, donde nombres en inglés que están escritos de manera distinta pero, con una pronunciación muy similar, se pueden obtener con una misma búsqueda de un código, como 'SOMERS' y 'SUMMERS' que ambos tienen el mismo código *Soundex* S562.

Tabla 2.2: Codificación de *Soundex* Americano [2].

Código	Letras
1	B, F, P, V
2	C, G, J, K, Q, S, X, Z
3	D, T
4	L
5	M, N
6	R

Aunque *Soundex* americano es eficiente y fácil de implementar, presenta varios problemas, tales como, su alta dependencia de la letra inicial, donde dos nombres que suenan idénticos son considerados distintos por el algoritmo si la primera letra es distinta (tales como "Korbin" y "Corbin") [10]. También presenta una baja intolerancia al ruido, donde una pequeña transposición de letras en un nombre en una base de datos (por ejemplo si el nombre está mal escrito por error), puede causar que el algoritmo no logre identificarlo en una búsqueda. Tampoco considera consonantes silenciosas, donde en inglés existen varios consonantes que en ciertos contextos no tienen sonidos pero la presencia de estas pueden afectar de gran manera el resultado del algoritmo (por ejemplo, "Coghburn" y "Coburn") [10]. Otro problema claro que tiene el algoritmo es que está enfocado solo al idioma inglés, ya que no considera letras de otros idiomas y varias de estas reglas no son consistentes con reglas fonéticas de otros idiomas. En particular, en el idioma español, no funciona bien porque no considera la letra

'ñ' en la tabla de codificación y tampoco considera las reglas fonéticas de las consonantes en español, por lo que, por ejemplo, la codificación de las palabras 'Halla' y 'Haya' en *Soundex* americano es H400 y H000, respectivamente. Es decir, el algoritmo los considera distintos aunque en español sean homófonos[3].

2.2.2. *Spanish Phonetic*

Existen variantes de *Soundex* orientados para codificar nombres propios en español, una de estas es *Spanish Phonetic* [3], el cual consiste en considerar las pronunciaciones de las consonantes en español y eliminar tanto la alta dependencia por la letra inicial del nombre, como la limitación de 4 caracteres en el código resultante [3]. El algoritmo de *Spanish Phonetic* sigue los siguientes pasos:

1. Escribir las palabras en mayúscula y descartar los signos de puntuación.
2. Eliminar las letras 'A', 'E', 'I', 'O', 'U', 'W' y 'H'.
3. Reemplazar las letras restantes del nombre por los valores de código que aparecen en la tabla 2.3.

Tabla 2.3: Codificación de *Spanish Phonetic* [3].

Código	Letras
0	P
1	B, V
2	F, H
3	T, D
4	S, Z, C, X
5	Y, LL, L
6	N, Ñ, M
7	Q, K
8	G, J
9	R, RR

2.2.3. *Metaphone*

Metaphone es un algoritmo propuesto por Lawrence Philips en 1990 [12], el cual consiste en realizar una codificación de una palabra basada en su pronunciación en inglés. Para esto, reemplaza consonantes con una codificación y mantiene las vocales en la codificación solo si están en el inicio de la palabra; similar a como lo hace el algoritmo *Soundex*. A diferencia de *Soundex*, *Metaphone* utiliza una serie de reglas más complejas que consideran los diptongos de las palabras y es capaz de retener más información al no dividir las letras en grupos como *Soundex* y sus variantes [13].

La codificación consiste en utilizar 16 consonantes: B, X, S, K, J, T, F, H, L, M, N, P, R, 0, W, Y, donde 'X' representa la consonante 'SH', '0' representa la consonante 'th' en inglés y el resto representa las consonantes a las que corresponden en inglés [11].

2.2.4. *Spanish Metaphone*

Spanish Metaphone (o algoritmo Metáfono para el español) es una adaptación de *Metaphone* orientada a las palabras en español, creada por Alejandro Mosquera [4]. Su principal diferencia con el *Metaphone* tradicional es que *Spanish Metaphone* considera los caracteres especiales en español, lo que incluye las letras 'Ñ', 'LL', las vocales con tildes, entre otros, donde unos ejemplos de remplazos, que se agregaron en esta variante de *Metaphone*, se encuentran en la tabla 2.4.

Tabla 2.4: Ejemplo de codificaciones para caracteres en español realizadas por Spanish Metaphone [4].

Letras	Reemplazando
á	A
ch	X
C	S
é	E
í	I
ó	O
ú	U
ñ	NY
ü	U
b	V
Z	S
ll	Y

2.3. Métricas de Cadena de Caracteres

Las Métricas de Cadena de Caracteres corresponden a métricas que calculan la distancia o similitud entre un par de palabras, utilizando distintos criterios [14]. Son ampliamente utilizados en la búsqueda de palabras, corrección de errores, entre otros [15].

2.3.1. Distancia de Levenshtein

La distancia de Levenshtein, nombrada tras Vladimir Levenshtein por su propuesta del algoritmo en 1965, calcula la distancia entre palabras como la cantidad mínima de adiciones, substracciones y substituciones necesarias para transformar una palabra en la otra [16].

2.3.2. Algoritmo ALINE

El algoritmo ALINE realiza un conjunto de operaciones sobre un par de palabras que considera la similitud entre pronunciación de las consonantes y vocales, no solo para medir la similitud del par de palabras, sino que para encontrar el segmento de cada palabras que mejor se alinea [5]. El algoritmo incluye las operaciones de inserciones/substracciones, substituciones y expansiones/reducciones, aparte de reemplazar las palabras con una aproximación de pronunciación fonética.

Una lista de ejemplos del funcionamiento de ALINE se puede observar en la tabla 2.5. En esta tabla se encuentra que la parte que se alinea mejor del par "three" y "trēs" es la parte correspondiente a "three" (que fonéticamente lo reescriben como "$\theta ri y$") con "trē"; en el par "blow", "flare" la parte que se alinea mejor es la parte correspondiente a "blo", "fla"; y en el par "tooth" y "dentis" es "tooth" (reescrito como "$tuw\theta$") con "tis".

Tabla 2.5: Ejemplo de resultado de aplicar el algoritmo de alineación ALINE a tres pares de palabras [5].

Par de palabras	Alineación de ALINE
three : trēs	$\|\|\ \theta\ r$ i$y\ \|\|$ $\|\|\ t\ r$ ē $\|\|\ s$
blow : flare	$\|\|\ b\ l\ o\ \|\|\ w$ $\|\|\ f\ l$ ā $\|\|$ re
tooth : dentis	$\|\|\ t\ uw\ \theta\ \|\|$ de$n\ \|\|\ t$ i $s\ \|\|$

2.3.3. Distancia Coseno

La Distancia de Coseno, o Coeficiente Ochiai es una forma de calcular la similitud entre vectores o conjuntos y, para la forma de conjuntos, sigue la fórmula que se ve en la ecuación 2.1 [17], la cual se puede aplicar para encontrar similitud de oraciones o palabras.

$$Coseno(x, y) = \frac{|X \cap Y|}{\sqrt{|X| \cdot |Y|}}. \tag{2.1}$$

2.4. Aprendizaje de Máquinas

El Aprendizaje de Máquinas, o *Machine Learning*, es una rama de la inteligencia artificial que se dedica al estudio de algoritmos que aprenden a predecir o clasificar conjuntos de datos de manera automática [18].

Los algoritmos de Aprendizaje de Máquinas se suelen dividir en cuatro etapas, tal como aparece en la figura 2.1: La entrada, corresponde a los conjuntos de datos a ingresar al algoritmos, los que pueden ser imágenes en los problemas de visión de computador o textos

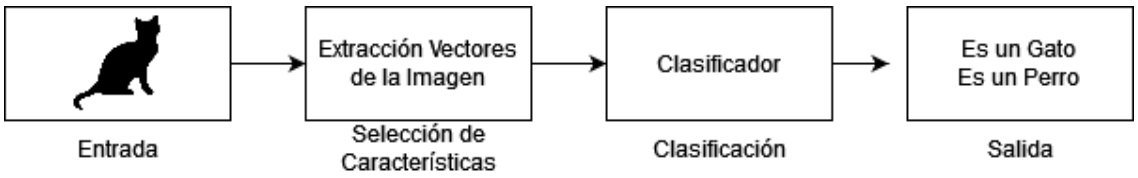


Figura 2.1: Diagrama Simplificado de las etapas dentro del Aprendizaje de Máquinas.

en el caso de procesamiento de lenguaje natural, entre otros. Luego, viene la Selección de Características, donde se eligen las características que se creen más importantes de la entrada para poder solucionar un problema, reduciendo así la complejidad de este. La tercera etapa es el Clasificador, que corresponde al algoritmo que es capaz de aprender a clasificar o predecir a partir de las características elegidas de las entradas. Por último, la Salida, que corresponde al resultado del modelo al recibir una entrada, puede ser una clase, instancia, predicción, etc [19].

2.5. Aprendizaje Profundo

El aprendizaje profundo o *Deep Learning*, son modelos de Aprendizaje de Máquinas que utilizan Redes Neuronales Artificiales con muchas capas ocultas, por lo que suelen ser algoritmos mucho más complejos que los clasificadores utilizados tradicionalmente en Aprendizaje de Máquinas. Esta complejidad les permite contener una parte o el total de la selección de características dentro de las capas ocultas de la red neuronal [20], tal como se muestra en el diagrama simplificado en la figura 2.2.

Las Redes Neuronales Artificiales son modelos de varias capas que reciben una entrada, donde cada capa está compuesta por nodos, que se denominan como neuronas, los cuales se componen de parámetros que extraen información importante de la entrada a la capa y se actualizan a medida que la red neuronal es entrenada [6].

Hay varias formas de entrenar una red neuronal, como el Aprendizaje Supervisado, que consiste en tener conjuntos de entrenamiento donde cada entrada está pre-etiquetada con los resultados esperados de ingresar la entrada a la red, donde los parámetros de la red se van actualizando al comparar las estimaciones de la red con las etiquetas esperadas. Existe un parámetro denominado tasa de aprendizaje que se elige para poder controlar la medida con la que se actualizarán los parámetros, donde a mayor tasa de aprendizaje los parámetros se adaptan más ante los errores durante las estimaciones [6].

Existen varios tipos de Redes Neuronales, como las Redes Neuronales Convolucionales, las cuales son muy populares para resolver problemas relacionados a imágenes y visión de computador; y las Redes Neuronales Recurrentes, las cuales son utilizadas en problemas donde importa la secuencia de los datos, como análisis de textos, análisis de vídeos, etc [20].

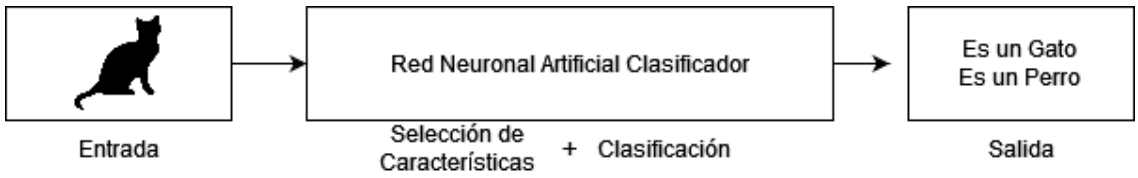


Figura 2.2: Diagrama Simplificado de las etapas dentro del Aprendizaje Profundo.

2.5.1. Redes Neuronales Convolucionales

Una Red Neuronal Convolucional es una red neuronal artificial que se utiliza en gran medida en el área de visión por computadores, por lo que suelen estar estructuradas para recibir como entrada una imagen. En general, su arquitectura se compone de las siguientes capas [6]:

- Capa de Entrada, la que corresponde a los pixeles de la imagen de entrada.
- Capas Convolucionales, donde hay filtros entrenables, denominados kernels, que se aplican a la entrada de la capa extrayendo características de esta. Estas características se guardan en las neuronas, las que se conectan parcialmente entre sí, donde se evita la conexión entre todas las neuronas para que el modelo no sea tan grande, lo que dificultaría el entrenamiento de este [6].
- Capa de Pooling, donde se realiza un sub-sampleo de la entrada, es decir, se reduce la dimensionalidad de la entrada para disminuir el número de parámetros y, por lo tanto, reducir la complejidad computacional del modelo.
- Capa de Clasificación, en la que se conectan todas las neuronas de estas capas y se crean las clases de salida del algoritmo.

Estas capas se pueden ver ejemplificadas en la imagen 2.3, donde se observa una Red Neuronal Convolucional muy simple que recibe como entrada una imagen la cual debe clasificar como un número entre 0 y 9.

2.5.2. Redes Neuronales Recurrentes

Las Redes Neuronales Recurrentes son arquitecturas de redes neuronales que pueden ser entrenadas para aprender de secuencias de datos, por lo que son ampliamente utilizadas para generar secuencias en diversas áreas como música, procesamiento de texto y capturas de vídeos [21].

Aunque una Red Recurrente suficientemente grande debería ser capaz de aprender cualquier dependencia de secuencias de datos. En la práctica las Redes Recurrentes estándares no son capaces de guardar información de las entradas pasadas por mucho tiempo. Una variante que soluciona este problema es el Long short-term memory (LSTM), el cual es una arquitectura de Red Neuronal Recurrente que utiliza una celda de memoria especial para retener mejor la información [21].

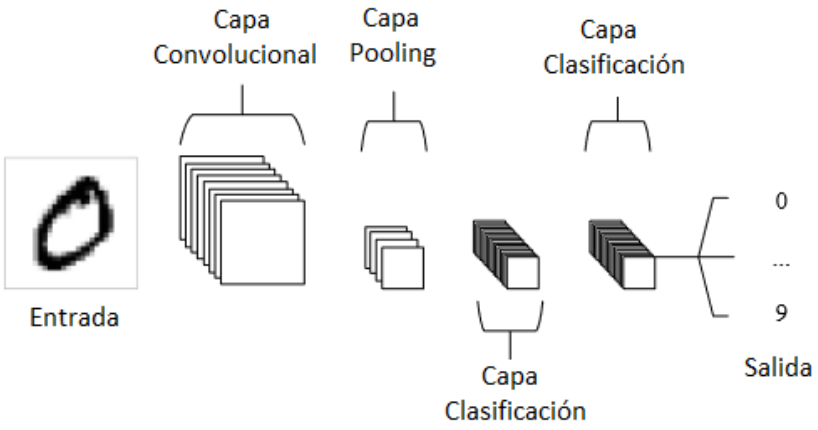


Figura 2.3: Ejemplo de Red Neuronal Convolucional simple de solo 5 capas. Basado en figura obtenida de [6].

La Red Siamesa LSTM es una arquitectura de Red Recurrente utilizada para clasificar pares de textos. Está compuesta de dos subredes LSTM idénticas, donde cada subred recibe como entrada un texto. La sub red LSTM extrae información de cada texto por separado y el resultado de cada sub red se junta en una función de energía para encontrar la similitud entre ellos, tal como se presenta en la figura 2.4, donde se ve una arquitectura Siamesa LSTM que analiza la similitud de oraciones para predecir similitudes semánticas, por lo que cada LSTM recibe como entrada una secuencia de palabras [7].

2.6. Podamiento de Filtros de Redes Neuronales

Una forma de acelerar las Redes Neuronales Convolucionales es podando o eliminando los parámetros que tienen menos efecto sobre el modelo completo por su baja contribución a la salida o por su redundancia, reduciendo así tanto el tamaño de la red como el núme-

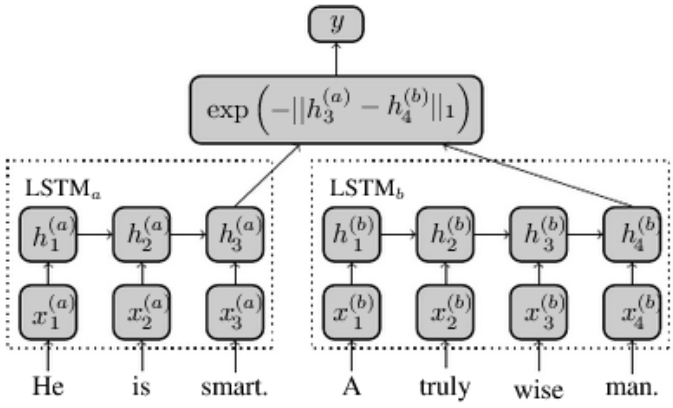


Figura 2.4: Ejemplo de Red Siamesa LSTM utilizado para detectar la similitud de oraciones. Figura obtenida de [7].

ro de cálculos dentro de esta. Al reducir el número de parámetros de la red también se suele producir una pérdida de precisión de la red, pero, dependiendo de cómo se aplica, se puede aumentar considerablemente la velocidad, manteniendo, o incluso hasta aumentando, levemente la precisión de la red [22].

Debido a que eliminar pesos específicos es complejo de implementar computacionalmente, los podamientos de Redes Neuronales Convolucionales se suelen enfocar en eliminar filtros completos dentro de las capas convolucionales, lo que elimina nodos completos con sus pesos [23].

Existen varias formas de elegir cuales son los filtros que menos aportan a la red, como rankear los pesos de cada filtro según su media, desviación estándar, norma L1 o L2, o el porcentaje de cero activaciones ante un conjunto de entradas [23].

Capítulo 3

Estado del Arte

Aunque las Redes Neuronales son ampliamente utilizadas para resolver problemas de procesamiento de texto, en el área de similitud de texto, la mayoría de las implementaciones en el estado de arte se enfocan en la semántica de los textos, es decir, en el significado de las palabras dentro de oraciones, como se puede observar en [24], [25], [26] y en [27], entre otros. Pero el problema que se quiere resolver en la presente memoria es la similitud de texto desde el punto de vista fonético, donde existe considerablemente menos literatura que utiliza redes neuronales.

Solo se lograron encontrar dos implementaciones que podrían solucionar el problema: una que trata de solucionar el problema de similitud para marcas en inglés y coreano [8], mientras que la otra busca resolver el problema de similitud de nombres de personas [9].

3.1. Generación de Características Fonéticas basadas en 2-Gram para una Red Neuronal Convolucional en la evaluación de similitud de marcas

Este método fue desarrollado con el propósito de medir la similitud de nombres de marcas en Corea, por lo que se enfoca en palabras en coreano y en inglés, utilizando una base de datos obtenida de Korea Intellectual Property Rights Information Service (KIPRIS) para entrenar la red neuronal [8].

Para la selección de características de los nombres de las marcas, primero romaniza las palabras coreanas, para que queden todas las palabras en el alfabeto latino, entonces transcribe las palabras al Alfabeto Fonético Internacional (IPA) y utiliza el algoritmo de alineamiento de texto ALINE para seleccionar los segmentos más similares del par.

Luego separa las segmentos resultantes en sub-segmentos de pares de letras denominado 2-gram, donde se crean los subconjuntos de dos letras consecutivas de las palabras. Por ejemplo, la palabra *"TEXT"* se separa en $-T$, $''TE''$, EX, XT y $T\_$, con $''-''$ un símbolo para marcar el inicio de la palabra y $''\_''$ un símbolo para marcar el final.

Entonces utiliza un diccionario creado por los autores que se encuentra en la figura 3.1 para transformar los conjuntos 2-gram en coordenadas 2D, donde se puede observar que fonemas similares tienen valores cercanos en el diccionario (como en la cuarta fila se observan valores similares para $''b''$ y $''v''$ o en las últimas dos filas se ve que la $''m''$ y la $''n''$ están muy cercanos).

Las coordenadas obtenidas se dibujan en una imagen de 128 x 128, pixeles conectando las coordenadas en orden con líneas rectas, donde las distintas marcas utilizan distintos canales de color (una palabra en rojo y la otra en verde). Los traslapes entre las figuras de cada palabra se ven en amarillo, con la intuición que más traslapes implican que la pronunciación de las palabras deben ser similares. La intensidad de cada línea va decrementando mientras más larga es la palabra original, donde la intensidad de cada línea G_{i} sigue la ecuación 3.1, donde Z es un factor de escala que eligieron con el valor de 255 y γ es un factor de descuento que eligieron, finalmente, como 0,9.

Finalmente, se obtiene una imagen como la que se ve en el ejemplo de la figura 3.2, la que se ingresa a la Red Neuronal Convolucional que se encuentra en la figura 3.3 para que esta estime la similitud de las líneas que representan las marcas.

$$PF(G_{\mathrm{i}}) = Z \prod_{k=o}^{\mathrm{i}} \gamma^{k}. \tag{3.1}$$

```
- : 16,  i : 19,  ɪ : 21,  y : 23,  e : 25,  ɛ : 27,
ə : 29,  ɜ : 30,  æ : 32,  a : 33,  ɑ : 35,  ʌ : 37,
ɔ : 38,  o : 40,  ʊ : 42,  w : 44,  u : 46,  h : 48,
p : 51,  b : 53,  v : 55,  f : 57,  c : 59,  k : 61,
q : 63,  g : 66,  d : 69,  t : 72,  θ : 74,  ð : 76,
s : 79,  ʃ : 81,  ʧ : 83,  x : 85,  z : 88,  ʒ : 90,
ʤ : 92,  j : 94,  r : 97,  ɹ : 99,  l : 101,  m : 104,
n : 106, ŋ : 108, _ : 111
```

Figura 3.1: Diccionario de fonemas en inglés en su representación de alfabeto fonético internacional a valores utilizado por el algoritmo basado en 2-gram. Valores obtenidos de [8].

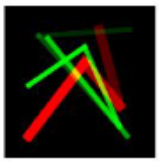
Entrada	Preprocesamiento	Imagen Resultante
grass : gramen	|| g r æ || s || g r a || men	

Figura 3.2: Ejemplo de Selección de Características de un par de palabras por el método basado en 2-Gram. Basado en figura obtenida en [8].

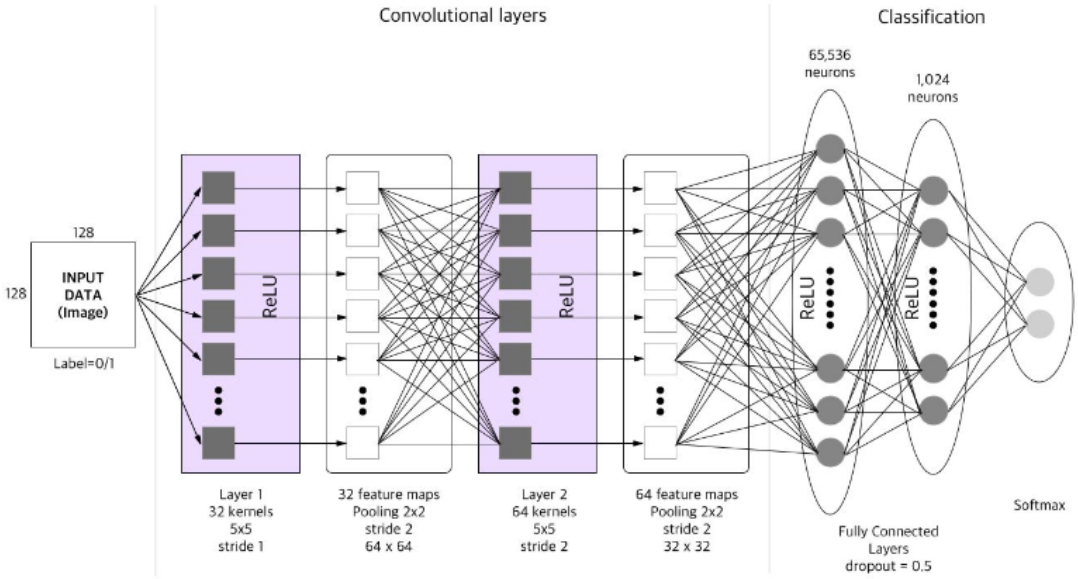


Figura 3.3: Red Neuronal Convolucional que utiliza el método basado en 2-Gram. Figura obtenida de [8].

3.2. Hello My Name Is (HMNI)

Es un método que utiliza técnicas de lógica difusa en conjunto a una red neuronal profunda Siamesa LSTM para medir la similitud fonética entre nombres de personas, con el objetivo de poder identificar si un nombre corresponde a un apodo de otro [9].

Para lograr este objetivo, crearon una base de datos con listas de nombres alternativos comunes, calcularon más de 20 características a través de diferentes métricas y algoritmos fonéticos, en conjunto con una Red Neuronal Recurrente y otros clasificadores de aprendizaje de máquinas [9].

Para la selección de características de los nombres, primero aplica al par de nombres un simple pre-procesamiento para limpiar los nombres (eliminar los caracteres especiales, cambiar las palabras a minúscula, etc.) y luego obtiene 5 resultados de algoritmos distintos con este par que utiliza como características antes de entregárselos a un clasificador.

Uno de las características que utiliza proviene de ingresar los nombres a una red neuronal Siamesa LSTM tal como se ve en la figura 3.5; otra características la obtiene ingresando el resultado 20 algoritmos de distancia distintos y 3 algoritmos de comparación de cadena de caracteres sobre el par de nombres a un clasificador Random Forest; el resto de las características elegidas corresponden a 3 de los algoritmos de distancia ya calculados anteriormente.

Para la clasificación ingresa las características 5 anteriores a un modelo Regresión Logística para obtener la similitud entre el par de nombres, tal como se muestra en la figura 3.4.

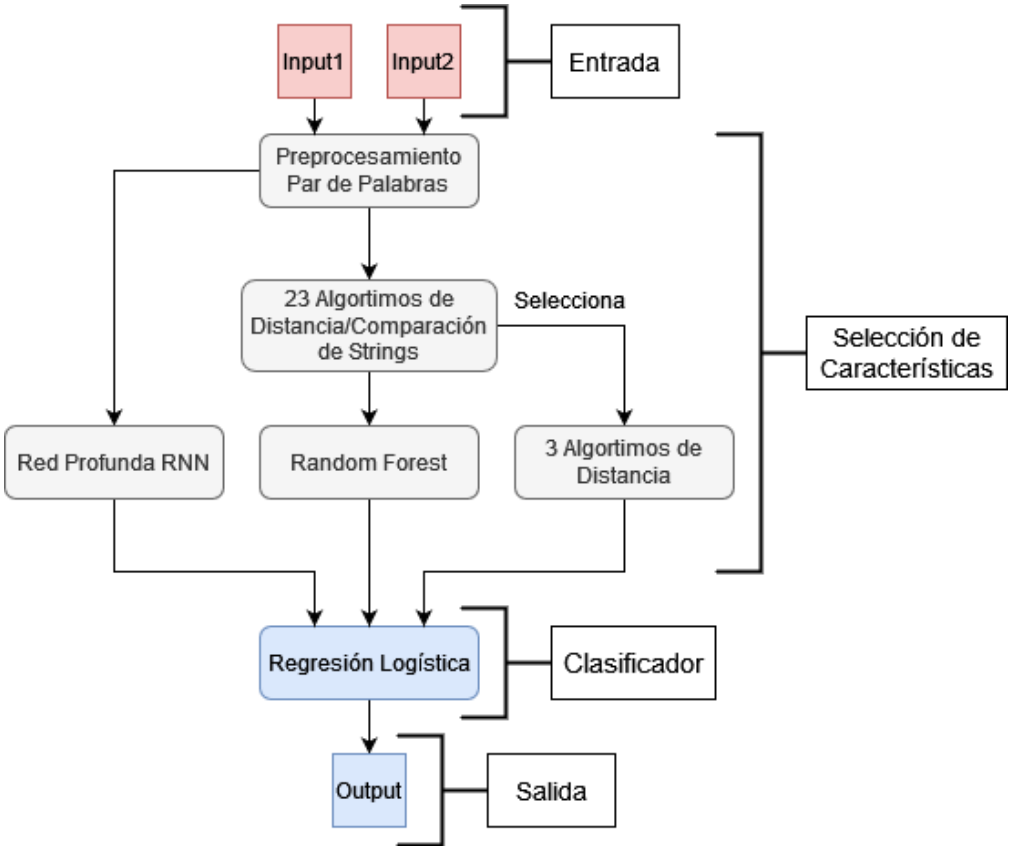


Figura 3.4: Diagrama del algoritmo HMNI.

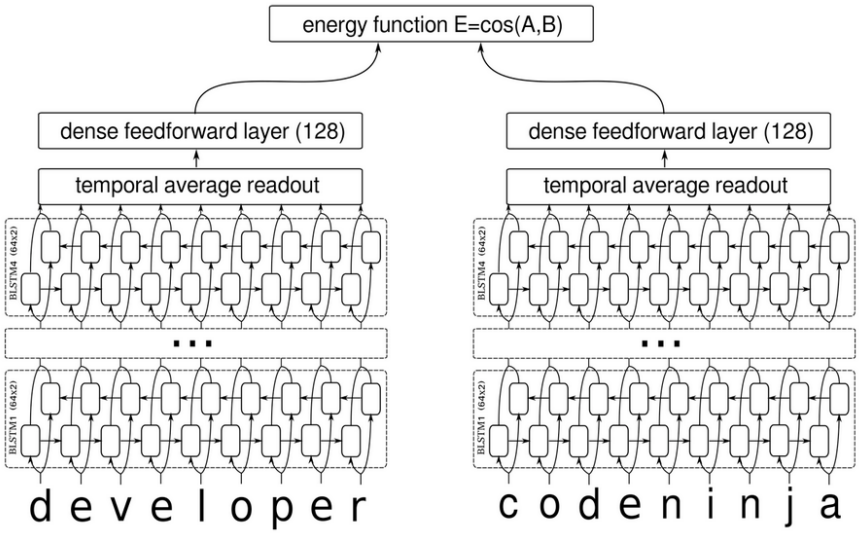


Figura 3.5: Red LSTM utilizada en HMNI con caracteres de entrada. Figura obtenida de [9].

Capítulo 4

Metodología

A continuación se explican las metodologías utilizadas para crear la Base de Datos que se usó para entrenar y comparar los algoritmos, implementar el Buscador Fonético y las métricas de comparación utilizadas en los distintos algoritmos.

4.1. Metodología para Creación y Extensión de la Base de Datos

Para crear la Base de Datos utilizada, primero, se crearon manualmente listas de pares de marcas que son fonéticamente similares. Para crear las listas de palabras, se utilizaron dos buscadores fonéticos de marcas gratuitos en línea: uno, corresponde a un buscador fonético de Marcanet, proporcionado por el Instituto Mexicano de la Propiedad Industrial para buscar marcas en la base de datos de la Dirección Divisional de Marcas de México [28]. El segundo, corresponde a un buscador fonético que busca marcas en una base de datos internacional de la Organización Mundial de la Propiedad Intelectual, proporcionado por la misma organización [29], el cual fue utilizado para extender las bases de datos durante la memoria.

Para armar las listas de pares de marcas se buscaron nombres de marcas en los buscadores fonéticos mencionados anteriormente, obteniendo largas listas de marcas similares fonéticamente a la marca buscada. Luego, las listas se redujeron eligiendo manualmente solo entre 5 y 10 marcas que se encontraron más similares a cada palabra buscada según su pronunciación en inglés o en español (si se encontraban menos de 5 marcas se buscaba otra marca). De esta manera, se obtuvieron 51 listas de marcas fonéticamente similares según sus pronunciaciones en inglés y 118 listas de marcas similares según sus pronunciaciones en español.

A partir de las listas de pares de marcas fonéticamente similares, se crearon combinaciones de pares de marcas, donde, los pares que que se originan de la misma lista se etiquetaron como clases positivas (fonéticamente similares); mientras que las combinaciones de marcas de distintas listas se etiquetaron como clases negativas (no son fonéticamente similares).

Se eligieron las combinaciones de tal manera que la proporción de clases negativas a

positivas sea 4 es a 1, proporción que se basó en cómo se realizó el entrenamiento en el algoritmo Hello My Name Is (HMNI)[9].

Por otro lado, también se utilizó la base de datos utilizada por el algoritmo HMNI que se encuentran en el mismo repositorio público donde se implementó el algoritmo [30]. Esta base de datos está compuesta de nombres de personas que son alias entre sí, los cuales son predominante nombres de culturas de habla inglesa aunque incluye nombres de varias culturas [9]. Se eliminó manualmente una porción de los emparejamientos originales de nombres que, aunque corresponden a aliases entre sí, se consideraron como no suficientemente fonéticamente similares entre sí, tales como "Richard" y "Dickson". Una vez limpiada la base de datos, se etiquetaron los pares como clases positivas y se generaron clases negativas, creando combinaciones de pares de nombres que son distintas a las originales.

Finalmente, estas Bases de Datos formadas se separaron en 60 % para entrenar los algoritmos, 20 % para validar los resultados del entrenamiento y el resto para realizar las pruebas de precisión de los algoritmos, proceso que se ve resumido en el diagrama de la figura 4.1 y que se realiza, por separado, para la lista de marcas fonéticamente similares en inglés, la lista de marcas fonéticamente similares en español y la lista de alias de nombres de personas, obteniendo así 3 Bases de Datos distintas.

4.2. Metodología para Cálculo de Precisión

La precisión de los algoritmos se midió bajo el criterio de Ranking Promedio, el cual es utilizado para medir la robustez de los algoritmos de Recuperación de Marcas (Trademark Retrieval) [31] y está dada por la ecuación 4.1, donde N es el número de marcas en el Conjunto de Prueba, N_{Rel} es el número de clases relevantes, que en este caso corresponden a las clases positivas, y R_{i} el ranking de la i-ésima clase positiva.

$$\widetilde{Rank} = \frac{1}{N \times N_{Rel}} \left(\sum_{\mathrm{i}=1}^{N_{Rel}} R_{\mathrm{i}} - \frac{N_{Rel}(N_{Rel}+1)}{2} \right). \tag{4.1}$$

Al estimar aplicar la ecuación 4.1, el mejor de los casos es cuando el Ranking Promedio da

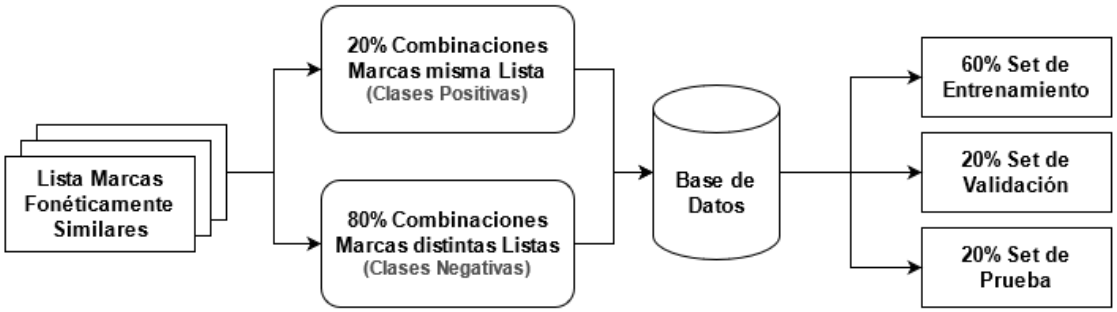


Figura 4.1: Diagrama de Metodología utilizada para crear la Base de Datos a partir de listas de marcas fonéticamente similares.

0, lo que significa que todos los primeros puestos están tomados por las clases positivas, mientras que el peor caso es cuando el algoritmo da 0.5, lo que implica que están completamente desordenadas las clases positivas y negativas [31].

Para comparar la precisión de los distintos algoritmos fonéticos para un conjunto de datos específico, se elige el conjunto de datos de prueba correspondiente dentro de la Base de Datos, se calcula la similitud de cada par del conjunto de datos de prueba utilizando el algoritmo, se rankean las similitudes obtenidas y se calcula el Ranking Promedio, tal como se ve en la figura 4.2.

4.3. Metodología del Buscador Fonético

El Buscador Fonético debe ser capaz de recibir una marca buscada, comparar su similitud fonética con cada marca dentro una base de datos de marcas y entregar una lista de marcas con alta similitud.

Para lograr esto se implementó un algoritmo que primero realiza una extracción de características tanto de la marca buscada como de cada marca en la base de datos de búsqueda (donde las características de la base de datos se pueden guardar para futuras búsquedas), parea las características de la marca buscada con cada una de las características extraídas de las marcas en la base de datos y se aplica alguno de los algoritmos fonéticos implementados para calcular la similitud de cada par.

Si la similitud de un par supera un cierto umbral, se agrega a una lista de entrega la marca correspondiente, tal como se aprecia en la figura 4.3.

4.4. Metodología para Cálculo de Velocidad

Para el cálculo de velocidad de los algoritmos se utiliza la metodología del buscador fonético, eligiendo una palabra buscada de manera arbitraria. La velocidad se calculó midiendo solo el tiempo promedio que se demoran los algoritmos de similitud en calcular la similitud de los pares de marcas ya preprocesados.

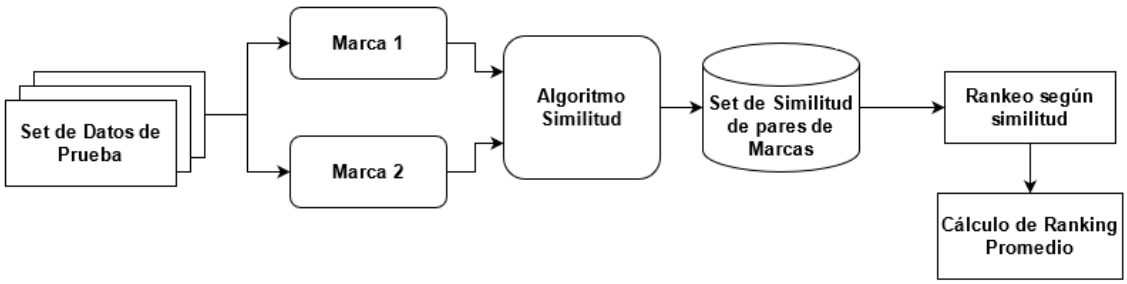


Figura 4.2: Diagrama de Metodología utilizada para medir la precisión de los algoritmos de similitud fonética.

Para el caso de las Redes Neuronales Artificiales, se utilizó la GPU de un computador laptop con una Tarjeta Gráfica GeForce RTX 2060.

La base de datos elegida para la estimación de velocidad fue una obtenida de la página web oficial de INAPI [32], obteniendo así las marcas registradas en esa Institución desde el año 2009 hasta el año 2020. Se eliminaron los nombres de marcas repetidas (recordar que INAPI acepta denominaciones de marcas repetidas con tal que no sufran similitud en su cobertura) y se obtuvo una base de datos de 292.706 marcas.

Cabe mencionar que no se estima el tiempo de extracción de características de las marcas debido a que el tiempo en extraer solo las características de la marca buscada suele ser ínfima en comparación con el resto de los procesos, mientras que las características ya extraídas de la base de datos a utilizar se puede guardar y cargar. Por lo tanto, en el uso cotidiano del buscador se espera que se preprocesen las bases de datos de antemano.

4.5. Preprocesamiento de símbolos y números

Para considerar posibles símbolos y números, no expresados en palabras de las marcas que pueden afectar la fonética de estas, se estudiaron cuáles son los caracteres especiales más utilizados en la base de datos de INAPI. Los resultados de este estudio están resumidos en la tabla 5.1 donde se encontró que los caracteres especiales '&', '+', '@' y '%' son frecuentes y pueden tener un efecto sobre la fonética de la palabra a la que pertenece.

En cuanto a las posibles pronunciaciones de los caracteres encontrados, se tiene que puede cambiar según el idioma o el contexto, por lo que, al esperarse que el público en Chile tienda a pronunciar las marcas más en español que en inglés, se elige corresponder la pronunciación de estos signos con palabras en español.

Considerando lo anterior, se realizó un bloque de preprocesamiento para todos los algoritmos implementados, donde se reemplazan los símbolos más frecuentes y los números por su versión en palabras en español según ejemplos encontrados en la base de datos de INAPI. Este bloque consiste en reemplazar $'\&'$ por 'y' como en las marcas "KREA HOGAR & TEXTIL", "ROCK & GUITARRAS" y "SCHNEIDER & MORALES"; $'+'$ por 'más' como en las

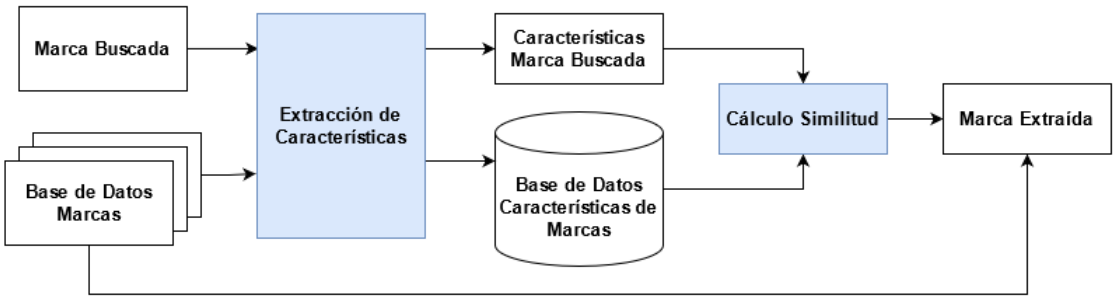


Figura 4.3: Diagrama de Metodología utilizada en los Buscadores Fonéticos, donde en azul se observa la extracción de características y el cálculo de similitud realizada por algún clasificador.

marcas "NO + CLAVOS", "+ CAFE AROMA Y SABOR" y "+VISION"; '@' por 'a' por marcas como "INTEGR@CCESO DE INTEGRAMEDICA", "DAT@BUSINESS" y "PANORAM@"; '%' por 'porciento' como en las marcas "CITRIC, 100 % EXPRIMIDO", "UNO AL DIA 0 %" y "CORTADOR 48 % SL". Por otro lado, se utilizó una librería de Python llamado num2words [33] para reemplazar los números en palabras en español.

4.6. Implementación de Algoritmos Tradicionales

Con el objetivo de poder comparar los algoritmos con redes neuronales implementadas, también se implementaron algoritmos tradicionales de comparación fonética. Para esto, se aplicó primero el bloque de preprocesamiento de símbolos y números a las palabras y se probaron cinco configuraciones de preprocesamiento: cuatro que aplican uno de los siguientes algoritmos de codificación fonética: *Soundex*, *Metaphone*, *Spanish Phonetics* y *Spanish Metaphone*; y una configuración que no aplica codificación. Luego, para calcular la similitud de la palabras ya preprocesadas, se prueba con dos configuraciones distintas: una aplicando la Distancia de Levenshtein y otra aplicando la Distancia de Coseno.

Para el cálculo de velocidad de estos algoritmos se estima solo en el tiempo promedio de cálculo de similitud del algoritmos de distancia sobre cada par de la base de datos de INAPI ya preprocesado según la configuración respectiva.

Para implementar la distancia de Levenshtein se utilizó la librería Python-Levenshtein [34], la cual corresponde a una implementación del algoritmo utilizando módulos de C, donde las implementaciones en C suelen ser más rápidas que las implementación de Python [35]. Mientras tanto la distancia de Coseno y el resto de algoritmos de codificación se implementaron utilizando la librería de Abydos en Python [36].

Para el caso de la distancia de Levenshtein, la librería de Python-Levenshtein [34] solo entrega el número de operaciones mínimo para transformar una palabra en otra. Como no entrega una medida de similitud o de distancia normalizada entre 0 y 1, como las redes neuronales y la distancia de Coseno de Abydos, se decidió hacer esto manualmente. Guiándose de la implementaciones del mismo algoritmo pero en Abydos [37], se divide el resultado de la distancia de Levenshtein con la longitud de la palabra más larga comparada, es decir, se obtiene la similitud de un par de palabras $w1$ y $w2$ con la fórmula en la ecuación 4.2.

$$\widehat{Levenshtein}(w1, w2) = \frac{Levenshtein(w1, w2)}{Max\big(largo(w1), largo(w2)\big)}. \tag{4.2}$$

4.7. Implementación de Algoritmo con Red Convolucional Basado en 2-gram

El algoritmo basado en 2-gram se implementó en Python replicando la descripción de éste en [8] y agregando al inicio el bloque de preprocesamiento de símbolos y números. Para el

preprocesamiento de los datos se utilizó la librería de eng-to-ipa [38] para transformar todas las palabras al alfabeto fonético internacional, asumiendo que todos tienen un origen en inglés (cabe destacar que no es necesario implementar técnicas de romanización de caracteres ya que las marcas deberían estar en el alfabeto latino según las normas de INAPI). Luego, se utilizó la implementación de n-gram de la librería Abydos [36] para obtener la cuantización 2-gram de las palabras y se utilizó un diccionario muy parecido al de [8] solo que agregando la letra "ñ" y, debido a que es muy similar a la letra "n" se le asignó el valor cercano y disponible de 107 (ver figura 3.1).

Al momento de dibujar las líneas de las imágenes, se aplicó la ecuación 3.1 para calcular la intensidad de las líneas pero, después de analizar los resultados obtenidos en 5.11, se decidió entrenar y probar 3 variaciones del algoritmo. Estas configuraciones tienen distintos valores de factores de descuentos para tratar de contrarrestar los efectos de la intensidad decreciente de las líneas correspondientes a las letras más alejadas del inicio de la palabra. Por último, la Red Neuronal Convolucional se implementó en Tensorflow [39], donde en [8] se incluye la lista de capas, parámetros y funciones requeridas para implementarlo en esa plataforma.

Cabe mencionar que durante el preprocesamiento, a diferencia de [8], se decidió no utilizar el algoritmo de alineamiento ALINE. Esto debido a que, para que la implementación sea eficiente, lo ideal es correr de antemano el preprocesamiento de la base de datos completa y guardar las marcas preprocesadas para futuros usos. Pese a lo anterior, el algoritmo ALINE requiere conocer la marca buscada de antemano, lo que implicaría que cada vez que se consulte una nueva palabra, en una misma base de datos, habría que recalcular parte del preprocesamiento, incrementando considerablemente el tiempo de cómputo sobre la base de datos completa. Por lo tanto, se eligió eliminar el uso de ALINE con el fin de independizar el preprocesamiento del procesamiento de la red neuronal .

4.8. Implementación de Red Recurrente Usada en HMNI

El Algoritmo de HMNI fue implementado en [9] usando Python en conjunto con librerías de Tensorflow [39], implementación que se encuentra disponible en un repositorio de código abierto [30]. Por lo tanto, se utilizó este repositorio como base y se realizaron algunas modificaciones.

La primera modificación fue agregar el bloque de preprocesamiento de símbolos y números en español antes de realizar el primer preprocesamiento del algoritmo original.

El otro cambio importante realizado fue modificar la Red Siamesa LSTM para evitar que re-calcule vectores redundantes al momento de funcionar el Buscador Fonético, aprovechando que la entrada correspondiente a la palabra buscada será constante. Para esto, durante la primera comparación de marcas, se guardan en un archivo externo los vectores de características procesados por la sub-red LSTM, correspondiente a la palabra buscada antes que llegue a la función de energía donde se compara con los vectores de la otra sub-red LSTM. De esta manera, en vez de re-calcular los vectores de la sub-red con la palabra buscada durante las siguientes comparaciones, se carga el archivo con los vectores guardados, ahorrando tiempo de procesamiento. El método anterior se puede apreciar en el diagrama de la figura 4.4.

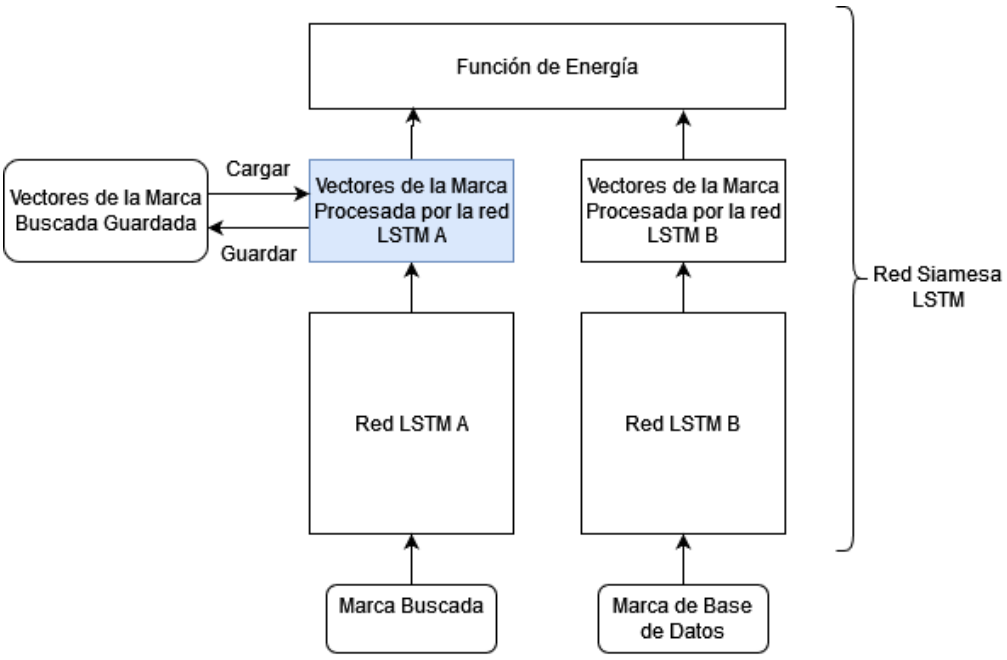


Figura 4.4: Diagrama de Red Siamesa LSTM donde se muestra la ubicación en la que se guardan y cargan los vectores procesados por la sub red LSTM que recibe como entrada a la marca buscada.

4.9. Entrenamiento de las Redes

Para realizar los entrenamientos de las redes, se utilizaron los conjuntos de entrenamiento y de validación creados con la metodología de la Base de Datos, donde la red aprende sus parámetros utilizando los ejemplos contenidos en el conjunto de entrenamiento, mientras que el conjunto de validación se utiliza para ir evaluando el ajustamiento de las redes entrenadas. Para implementar los entrenamientos se utilizaron las funciones de entrenamiento automático respectivas de Tensorflow, utilizando funciones de optimización por defecto y una tasa de aprendizaje inicial de 0,001, donde, cada cierta cantidad de iteraciones se disminuye la tasa, llegando hasta una tasa de 0,000001.

Debido al alto uso de GPU durante el entrenamiento, se decidió correr esto dentro de una cuenta gratuita en Google Colab [40], el cual es una nube de google creada para ejecutar código en Python. Pero, debido a que se utilizó su versión gratuita, presentó límitaciones en cuanto al tiempo de operación por lo que los entrenamientos debieron hacerse en segmentos discontinuos, guardando los parámetros resultantes de cada segmento de entrenamiento y cargando estos en el siguiente.

4.10. Metodología de Podamiento de las Redes Neuronales Convolucionales

Una vez que la Red Neuronal Convolucional fue entrenada, se procedió a realizar distintos podamientos de los pesos que aportan menos a la red. Para esto, primero se calculó la norma l1 de los filtros de la red entrenada, por lo que a cada filtro i de cada capa l dentro de las capas convolucionales de la red, se le aplica la fórmula que aparece en la ecuación 4.3, donde $k_{\mathrm{i},j}$ son los pesos de los filtros [23].

$$s_{l,\mathrm{i}} = \frac{1}{n} \sum_{j=1}^{n} |k_{\mathrm{i},j}|. \tag{4.3}$$

Entonces, se eliminan los filtros con menor norma l1 hasta alcanzar cuotas correspondientes a porcentajes de la cantidad total de filtros. Para alcanzar velocidades que se puedan distinguir se decidió hacer 5 podamientos distintos, eligiendo los porcentajes correspondientes a 10 %, 25 %, 35 %, 50 % y 65 %. Luego de podar las redes se procedió a hacer de nuevo un corto entrenamiento en el mismo conjunto de datos de entrenamiento con el que fue entrenada la red base.

Para implementar el podamiento de la red se basó en las instrucciones de un post que explica un código para podar Redes Convolucionales genéricas implementadas en Tensorflow de Python, utilizando la librería Keras-surgeon [41] para ejecutar los podamientos [23].

Capítulo 5

Resultados

5.1. Estudio sobre la Base de Datos de INAPI

Se realizó un estudio sobre la Base de Datos obtenida de INAPI de 292.706 marcas registradas, donde se obtuvieron todos los distintos caracteres especiales que no corresponden al alfabeto latino, lo que incluye signos de preguntas, puntuaciones, entre otros, para así poder analizar cuales de los signos más comunes pueden tener un efecto sobre la fonética de los nombres de algunas marcas.

Parte de los resultados de este estudio están en la tabla 5.1, encontrándose todos los caracteres especiales que tienen una frecuencia de al menos un 0.5 % sobre el total de marcas en la Base de Datos y se puede observar que los caracteres especiales más comunes que pueden tener un efecto sobre la fonética de la palabra a la que pertenece son: '&', '+', '@' y '%'. Para ver el estudio completo con todos los caracteres, ver apéndice A.1, A.2 y A.3.

5.2. Extensión de la Base de Datos

Originalmente, se tenían dos bases de datos con marcas elegidas de manera manual, utilizando el buscador fonético de Marcanet [28]: una con marcas fonéticamente similares según su pronunciación en inglés y otra con marcas similares según su pronunciación en español. Estas dos bases de datos se extendieron durante la memoria utilizando el buscador fonético proporcionado por la Organización Mundial de la Propiedad Intelectual para encontrar más listas de palabras fonéticamente similares. Se obtuvieron 51 listas de palabras fonéticamente similares en inglés, lo que se tradujo en 1272 clases positivas y 118 listas de palabras fonéticamente similares en español, lo que se tradujo en 3074 clases positivas.

Por otro lado, también se agregó la base de datos de alias de nombres utilizada por [9] para entrenar su red, la cual contenía 17.510 pares de aliases de nombres, donde, al eliminar los pares de nombres que claramente no tienen similitud fonética, quedaron solo 13.612 clases positivas.

Tabla 5.1: Caracteres especiales más comunes encontrados en la Base de Datos de 292.706 marcas registradas en la INAPI y sus respectivos porcentajes.

Carácter Especial	Porcentaje en la Base de Datos de INAPI
.	43.1 %
-	30.1 %
,	16.6 %
&	12.5 %
’	10.0 %
!	3.9 %
+	2.8 %
”	1.7 %
´	1.0 %
:	1.0 %
/	0.9 %
)	0.9 %
(	0.8 %
’	0.6 %
@	0.5 %
%	0.5 %
?	0.5 %

El número de pares de datos totales de las bases de datos creadas (clases positivas y negativas) se encuentran resumidos en la tabla 5.2, donde se observa que la base de datos en inglés se logró casi duplicar. Se agregó también una base de datos de alias de nombres mucho más grande que las otras dos bases de datos y la base de datos en español se incrementó en menor medida, agregándose 1.020 pares.

A partir de estas bases de marcas se obtuvieron 1.272 pares de marcas en el conjunto de prueba en inglés, 3.074 pares de marcas en el conjunto de prueba en español y 13.612 pares de nombres en el conjunto de prueba de alias de nombres, los que fueron utilizados para realizar las pruebas de precisión de los distintos algoritmos. Mientras que el resto de pares de marcas y palabras fue utilizado para entrenar las redes en sus distintas configuraciones.

Tabla 5.2: Tamaño de las Bases de Datos Original en comparación con las Bases de Datos extendidas durante el Trabajo de Título.

Base de Datos	Cantidad de Pares de Marcas Originales	Cantidad de Pares de Marcas Extendida
Inglés	3.855	6.360
Español	14.250	15.370
Alias de Nombres	-	68.060

5.3. Comparación de Velocidad entre los algoritmos sin Podamiento

Se utilizó la metodología de cálculo de velocidad, tanto con las redes neuronales artificiales como con los algoritmos tradicionales de distancia implementados, donde se buscó la palabra "venta" (la que fue elegida de manera completamente arbitraria) dentro de la base de datos obtenida de INAPI.

Los resultados de aplicar esta metodología se encuentran en la tabla 5.3, donde se observa que el algoritmo más rápido es el de la Distancia de Levenshtein, con 0.10 ms por cada par o alrededor de 30s en realizar la búsqueda fonética en la base de datos completa, seguido por la distancia de Coseno, alcanzando una velocidad de 0.18 ms por par, la cual es muy cercana la velocidad de la Red Convolucional del algoritmo basado en 2-Gram con 0.20 ms por par utilizando GPU. Estos tres resultados son mucho más rápido que la red recurrente de HMNI, la cual, incluso cargando los vectores guardados de la sub-red LSTM de la marca buscada, se demora 27 ms por par con GPU, es decir, más de 2 horas en realizar una búsqueda fonética completa dentro de la base de datos de INAPI.

Tabla 5.3: Tiempo promedio en procesar una búsqueda fonética con distintos algoritmos dentro de la base de datos de INAPI de 292.706 marcas.

Algoritmo	Tiempo Promedio de Procesamiento
Distancia de Levenshtein	0.10 ms/par
Distancia de Coseno	0.18 ms/par
Red Convolucional de Algoritmo basado en 2-Gram	0.20 ms/par
Red Recurrente de HMNI Guardando Vectores	27 ms/par
Red Recurrente de HMNI Sin Guardar Vectores	56 ms/par

5.4. Comparación de Precisión entre Métodos Tradicionales

Se calculó el ranking promedio de las distintas configuraciones de Algoritmos Tradicionales sobre los Conjuntos de Prueba de cada base de datos implementada, obteniendo las precisiones que se ven en las tablas 5.4 y 5.5.

En la tabla 5.4 se encuentran los resultados de precisión de la Distancia de Levenshtein con sus distintas configuraciones y se observa que el peor resultado de precisión se obtuvo cuando no se aplica un algoritmo de codificación en el preprocesamiento, el cuál tiene los peores ranking promedios en los conjuntos de prueba en inglés y en español. El segundo peor resultado se obtuvo al aplicar *Soundex* en el preprocesamiento, resultando en la menor precisión en el conjunto de alias de nombres. En cambio, el mejor resultado de precisión se

obtuvo al aplicar la codificación *Metaphone*, resultando en el mejor ranking promedio en el conjunto de prueba en inglés y el de alias de nombre. Mientras que con la codificación *Spanish Phonetic* se obtuvo el mejor ranking promedio en español.

Por otro lado, en la tabla 5.5 se encuentran los resultados de precisión de las distintas configuraciones probadas con la Distancia de Coseno, donde se observan resultados análogos a los obtenidos con Levenshtein con variaciones muy pequeñas en los ranking promedios. Comparando los resultados en *Metaphone*, el cual entrega los mejores resultados en ambas tablas, las precisión obtenida por Levenshtein en el conjunto de prueba en español es levemente mejor, mientras que el resto es igual.

Tabla 5.4: Ranking Promedio sobre los conjunto de datos de Prueba de datos en Inglés, datos en Español y Alias de Nombres para el algoritmo de Distancia de Levenshtein utilizando distintas codificaciones en el preprocesamiento.

Algoritmo	Ranking Promedio Conjunto Inglés	Ranking Promedio Conjunto Español	Ranking Promedio C. Alias de Nombres
Levenshtein	0.19	0.21	0.04
Soundex + Levenshtein	0.14	0.16	0.13
Metaphone + Levenshtein	0.05	0.08	0.03
Spanish Phonetic + Levenshtein	0.07	0.07	0.05
Spanish Metaphone + Levenshtein	0.07	0.09	0.04

Tabla 5.5: Ranking Promedio sobre los conjuntos de datos de Prueba de datos en Inglés, datos en Español y Alias de Nombres para el algoritmo de Distancia de Coseno utilizando distintas codificaciones en el preprocesamiento.

Algoritmo	Ranking Promedio Conjunto Inglés	Ranking Promedio Conjunto Español	Ranking Promedio C. Alias de Nombres
Coseno	0.15	0.18	0.04
Soundex + Coseno	0.14	0.15	0.12
Metaphone + Coseno	0.05	0.09	0.03
Spanish Phonetic + Coseno	0.06	0.07	0.04
Spanish Metaphone + Coseno	0.09	0.11	0.04

5.5. Comparación Precisión de Algoritmo Basado en 2-Gram para los Conjuntos de Entrenamiento

Se entrenó la Red Convolucional con distintas combinaciones de conjuntos de entrenamiento: uno solo con el conjunto de entrenamiento en inglés, otro solo con el conjunto de entrenamiento en español, otro con ambos conjuntos y un último con ambos conjuntos y además, el conjunto de alias de nombres.

Los resultados de estos entrenamiento se encuentran en la tabla 5.6, observándose que los entrenamientos con el conjunto en inglés, español y con alias de nombres obtuvieron sus mejores precisiones en el conjunto de prueba respectivo al conjunto con el que se entrenaron pero obtuvieron bajas precisiones en el resto. Por otro lado, se observa que el resultado del entrenamiento en los conjunto de datos combinados en inglés y en español, obtuvo los mejores resultados en los conjuntos de prueba en inglés y en español, además de un muy buen resultado en el de Alias de nombres. Mientras tanto, la red entrenada en los tres conjuntos de entrenamiento combinados, obtuvo el mejor resultado en el conjunto de prueba de alias de nombres.

Tabla 5.6: Ranking Promedio sobre los conjunto de datos de Prueba de marcas en Inglés, marca en Español y Alias de Nombres para el algoritmo basado en 2-Gram entrenada en distintos conjunto de datos de entrenamiento.

Algoritmo	Ranking Promedio Conjunto Inglés	Ranking Promedio Conjunto Español	Ranking Promedio C. Alias de Nombres
Red 2-Gram Entrenada en Conjunto Inglés	0.08	0.25	0.20
Red 2-Gram Entrenada en Conjunto Español	0.30	0.14	0.27
Red 2-Gram Entrenada en D. Alias Nombres	0.16	0.19	0.04
Red 2-Gram Entrenada en D. Español + Inglés	0.04	0.06	0.04
Red 2-Gram Entrenada en D. Español + Inglés + Alias de Nombres	0.06	0.08	0.03

5.6. Comparación de Precisión de Algoritmo Basado en 2-Gram para distintos Factores de Descuento

Para tratar de evitar el efecto de la intensidad decreciente de las líneas dibujadas, que representan a las palabras en el preprocesamiento del algoritmo basado en 2-gram, se probó entrenar desde cero a la red con dos configuraciones de cálculos de intensidad distintos, uno con un factor de descuento de 0.95 y el otro con un factor de 0.99. Los efectos de elegir estos factores de descuentos sobre el valor original de 0.9 se encuentran resumidas en la tabla 5.7 donde se observa que, para un factor de descuento de 0.9, desde el onceavo valor en adelante los valores de la matriz se vuelven 0, es decir, desde el carácter onceavo en adelante la intensidad de la línea será 0. Mientras tanto, para un factor de descuento de 0.5 se observa que hasta el catorceavo carácter, la intensidad de la línea sigue siendo distinta de 0, aunque llega a un valor muy bajo, mientras que para un factor de descuento de 0.99 se mantiene con una intensidad muy alta; incluso hasta el catorceavo carácter. La tabla completa con todos los valores de intensidad se encuentra en el apéndice B.1.

Por otro lado, se entrenaron ambas configuraciones con los conjuntos de entrenamiento en inglés y español y se compararon los resultados de estas redes con la red entrenada con el valor original de factor de descuento de 0.9, resultados que se encuentran en la tabla 5.8, donde se observa que los mejores resultados de precisión en los tres conjunto de datos de prueba se obtuvieron con un factor de descuento de 0.9, mientras que el peor se obtuvo con un factor de descuento de 0.95.

Tabla 5.7: Primeros 14 valores de la matriz de Intensidad de líneas para 3 factores de descuento distintos elegidos.

Factor de Descuento	Primeros 14 valores de Intensidad de Líneas
0.90	[255, 229, 185, 135, 88, 52, 27, 13, 5, 2, 0, 0, 0, 0, ...]
0.95	[255, 242, 218, 187, 152, 118, 86, 60, 40, 25, 15, 8, 4, 2, ...]
0.99	[255, 252, 247, 240, 230, 219, 206, 192, 177, 162, 146, 131, 116, 102, ...]

Tabla 5.8: Ranking Promedio sobre los Conjunto de Prueba de datos en Inglés, datos en Español y Alias de Nombres para el algoritmo basado en 2-Gram entrenada con distintos factores de descuento.

Algoritmo	Ranking Promedio Conjunto Inglés	Ranking Promedio Conjunto Español	Ranking Promedio C. Alias de Nombres
Red 2-Gram Factor de Desc. de 0.90	0.04	0.06	0.04
Red 2-Gram Factor de Desc. de 0.95	0.08	0.12	0.25
Red 2-Gram Factor de Desc. de 0.99	0.07	0.10	0.23

5.7. Comparación de Precisión y Velocidad de Algoritmo Basado en 2-Gram para distintos podamientos

Luego de entrenar el algoritmo 2-gram en los conjuntos en español y en inglés, utilizando el factor de descuento de 0.9, se realizaron cinco podamientos de filtros, eliminando un 10 %, 25 %, 35 %, 50 % y 65 % de los filtros de las capas convolucionales con menor normal l1, respectivamente.

Los resultados de precisión y de velocidad de cada podamiento realizado, se encuentran en la tabla 5.9, donde se observa que, a medida que incrementa el porcentaje de filtros podados, disminuye tanto el tiempo promedio de procesamiento como la precisión de la red. La red sin podar resultó ser la más lenta con 0.204 ms/par, lo que se traduce en cerca de 1 minuto en analizar la base de datos de INAPI y la más rápida es la que tiene un podamiento de 65 %, con una velocidad de 0.135 ms/par, o cerca de 40 segundos en la base de datos. Se nota también que la precisión en los conjuntos de prueba en inglés y de alias de nombres se mantiene al podar solo un 10 % de los filtros, disminuyendo levemente la precisión en el conjunto de prueba en español. Luego, al comparar el filtro de 25 % con el de 10 % se tiene que se mantienen dos de las tres precisiones, disminuyendo solo la de alias de nombres. Comparando el podamiento de 35 % con el de 25 % se observa solo un decremento en el conjunto de pruebas en inglés. En el filtro de 50 % disminuyeron levemente las tres precisiones en comparación con el filtro de 35 % y, finalmente, para el caso de 65 % de filtros se observa una disminución sustancial de la precisión en los tres conjuntos de prueba en comparación con los todos los casos anteriores.

Tabla 5.9: Ranking Promedio sobre los Conjuntos de Prueba de datos en Inglés, datos en Español y Alias de Nombres para el algoritmo basado en 2-Gram, utilizando distintos porcentajes de podamiento.

Algoritmo	Ranking Promedio Conjunto Inglés	Ranking Promedio Conjunto Español	Ranking Promedio Conjunto Alias de Nombres	Tiempo Promedio de Procesamiento
Red 2-Gram Sin Podar	0.04	0.06	0.04	0.204 ms/par
Red 2-Gram 10 % Filtros Podados	0.04	0.07	0.04	0.182 ms/par
Red 2-Gram 25 % Filtros Podados	0.04	0.07	0.05	0.171 ms/par
Red 2-Gram 35 % Filtros Podados	0.05	0.07	0.05	0.164 ms/par
Red 2-Gram 50 % Filtros Podados	0.07	0.10	0.08	0.154 ms/par
Red 2-Gram 65 % Filtros Podados	0.26	0.29	0.27	0.135 ms/par

5.8. Resultados Cualitativos del Algoritmo Basado en 2-Gram

Luego de entrenar el algoritmo 2-gram en el conjunto en español más inglés, utilizando un factor de descuento de 0.9, se estimó la precisión de la red en los tres conjunto de prueba. A partir de esto, se eligieron ejemplos de los mejores resultados obtenidos, resultados que se encuentran en la tabla 5.10. También se eligieron ejemplos en el que el resultado no funcionó como se esperaba, los que se anotaron en la tabla 5.11.

En la tabla 5.10 se observan varios pares de palabras que fueron etiquetados como fonéticamente similares y la red fue capaz de estimar la similitud de manera completamente correcta (similitud 1.0). En esta tabla se ven ejemplos de pares de palabras que son fonéticamente similares según su pronunciación en inglés, como "SUGAR" con "CHUGAZ", y "FORTUNADO" con "TURN-AID-O". También se observan ejemplos de pares de palabras que son fonéticamente similares según su pronunciación en español, tales como "Hielo" con "YELLOW-SEA" y "+KOTA" con "DON MASCOTON". Asimismo, se encuentran ejemplos de alias de nombres que son fonéticamente similares en español e inglés, como "CATE" con "KATHY" y "LIZA" con "ALICE".

Tabla 5.10: Ejemplos de Resultados Cualitativos donde la Red Convolucional del algoritmo basado en 2-Gram obtiene buenos resultados de similitud estimada ante clases positivas.

Palabra 1	Palabra 2	Conjunto de Prueba de origen	Similitud Estimada
SUGAR	CHUGAZ	Inglés	1.0
KROSTY	BEFROSTIX	Inglés	1.0
FORTUNADO	TURN-AID-O	Inglés	1.0
UBER	UVEX	Español	1.0
HIELO	YELLOW-SEA	Español	1.0
+KOTA	DON MASCOTON	Español	1.0
INVERSIÓN	ESMAS DIVERSION	Español	1.0
CATE	KATHY	Alias de Nombres	1.0
LIZA	ALICE	Alias de Nombres	1.0

Mientras tanto, en la tabla 5.11 se observan ejemplos de pares que claramente presentan similitud fonética, pero la red falló por completo en detectarlas (dándoles similitud 0), tales como "TROTTER" que está literalmente contenido en la marca "GLOBETROTTER" o "TENS-HYL" que también está casi contenido en "PHOTO STENCIL" solo que con un par de letras distintas.

Tabla 5.11: Ejemplos de Resultados Cualitativos donde la Red Convolucional del algoritmo basado en 2-Gram obtiene malos resultados de similitud estimada ante clases positivas.

Palabra 1	Palabra 2	Conjunto de Prueba de origen	Similitud Estimada
TROTTER	GLOBETROTTER	Español	0.0
HIDROGEMA	ATRO UNGENA	Español	0.0
MI BURGER	WILLBURG	Español	0.0
TENS-HYL	PHOTO STENCIL	Español	0.0
ECONOFINANZAS	KONFIOPAY	Español	0.0
SOPROLE	BONPROLE	Español	0.0

Capítulo 6

Discusión de Resultados

Respecto a la extensión de base de datos que se encuentra en la tabla 5.2, la base en español se incrementó en bastante menor medida que la base de datos en inglés. Esto se debió a que al agregar más listas de marcas se debe verificar que estos no sean similares a otras listas ya creadas, o sino las clases negativas pueden terminar conteniendo marcas que si son fonéticamente similares, por lo que al tener más de 100 listas de marcas se hizo cada vez más difícil hacer esto de manera manual.

En cuanto a la comparación de velocidades de los distintos algoritmos que se encuentran en la tabla 5.3, se tiene que la Red Recurrente Siamese LSTM de algoritmo HMNI se demora tanto en comparación a los demás algoritmos debido a su alta complejidad donde, incluso eliminando la necesidad de recalcular los vectores de una de sus sub-redes LSTM, no fue suficiente para alcanzar velocidades aceptables, demorándose horas en un proceso que se quiere realizar lo más cercano a 30 segundos. Por lo tanto, se decidió enfocar los esfuerzos en mejorar la Red Convolucional del algoritmo basado en 2-Gram, el cual obtuvo una velocidad mucho más cercana a los algoritmos tradicionales. Por otro lado, el hecho que los métodos más rápidos correspondieron a los algoritmos tradicionales, es consistente con la simpleza de estos métodos en comparación con las redes neuronales. El algoritmo de Levenshtein fue el más rápido, con una velocidad de 10 ms por par, gracias al hecho que su implementación utilizaba código en C, lo cual le permitió alcanzar mayores velocidades que los códigos en Python, y se convirtió en la velocidad objetivo de los algoritmos implementados. Debido a que la velocidad de la Red Convolucional del algoritmo basado en 2-Gram estaba cercana a esta velocidad objetivo, es que se realizó un estudio del efecto de realizar distintos podamientos a las capas convolucionales para tratar de alcanzar una velocidad óptima.

Analizando los resultados de precisión de los algoritmos tradicionales de las tablas 5.4 y 5.5, se tiene que, para ambos algoritmos de distancia, los peores resultados se obtuvieron cuando no se utilizó algoritmos fonéticos en el preprocesamiento y cuando se usó *Soundex*. Lo primero muestra que la codificación fonética suele ayudar a los algoritmos de distancia a encontrar mejores similitudes. Mientras tanto, lo segundo es consiste con el hecho que *Soundex* es uno de los algoritmos de codificación fonética más simples y presenta varios problemas, como dar mucha prioridad a la primera letra de la palabra, tener una limitante de solo cuatro caracteres de codificación, entre otros. Por otro lado, usar la variante de *Soundex* en

español, *Spanish Phonetic*, resultó en mejores precisiones que con *Soundex*, lo que se debe a que esta versión trata de eliminar varias de las debilidades del algoritmo original. Los mejores resultados se obtuvieron con el preprocesamiento de *Metaphone*, lo que se debe a que *Metaphone* pierde menos información en su procesamiento que *Soundex* y sus variantes [13]. Por otro lado, *Spanish Metaphone* obtuvo una precisión levemente mejor en el conjunto de prueba en español, aunque peor en el resto, lo que es consistente con el hecho que este es una variante de *Metaphone*, enfocada al español. Por otro lado, comparando los resultados de la Distancia de Levenshtein con la de Coseno, se tiene que ambos rindieron de manera muy similar, obteniéndose que la configuración *Metaphone* más Levenshtein entregó los mejores resultados de precisión dentro de los algoritmos tradicionales.

Comparando las precisiones de la Red Neuronal Convolucional entrenada con los distintos conjuntos de entrenamiento que se encuentra en la tabla 5.6, se observaron bajas precisiones al entrenar solo con un conjunto de entrenamiento, por lo que se concluye que ningún conjunto de datos de entrenamiento creados es suficiente para que la red generalice en los otros conjuntos. En cambio, entrenar la red con más de un conjunto a la vez, le permitió a la red obtener mejores resultados en los tres conjuntos que al entrenar con solo uno, lo que implica que entrenar con los distintos conjuntos le ayuda a generalizar mejor. Mientras tanto, los resultados obtenidos con la red entrenada solo en los conjuntos de entrenamiento en español y en inglés, entregó mejores resultados que el entrenamiento con los 3 conjuntos de datos, a excepción del conjunto de prueba de alias de nombres donde se obtuvo un ranking promedio levemente más alto. Lo anterior, muestra que solo entrenar en esos dos conjunto de datos es suficiente para que la red generalice con el conjunto restante, lo que puede significar que el conjunto de prueba de alias de nombre sigue reglas de pronunciación sencillas. Esto último concuerda con el hecho que la mayor parte configuraciones de algoritmos tradicionales también presentan precisiones muy altas para ese conjunto de prueba, lo que se observa en las tablas 5.4 y 5.5.

Comparando las precisiones de la Red Convolucional del algoritmo basado en 2-Gram, entrenada, con inglés y español, con la Distancia de Levenshtein, utilizando *Metaphone*, se puede ver que la precisión obtenida con la Red Convolucional en los conjuntos de prueba en español e inglés, resultó mayor que los obtenidos por el método tradicional, aunque obtuvo una precisión levemente menor en el conjunto de prueba de alias de nombres. Por lo tanto, pese a que se demora el doble de tiempo en procesar los datos, la Red Convolucional logró obtener mejores resultados de precisión que los métodos tradicionales.

Observando los resultados cualitativos del algoritmo basado en 2-Gram, que se encuentran en las tablas 5.10 y 5.11, se observa que, pese a que la red logró obtener los mejores resultados de precisión, de todos modos falló en pares de palabras aparentemente fáciles, tales como "TROTTER" con "GLOBETROTTER" o "TENS-HYL"con "PHOTO STENCIL". Una posible causa de esto es el factor de reducción a las líneas que se van agregando al dibujo durante el preprocesamiento, lo que implica que letras más alejadas del inicio presentan líneas más tenues que las iniciales. Por lo tanto se decidió probar diferentes factores de reducción durante la metodología.

En cuanto a los efectos de probar diferentes factores de descuento durante el preprocesamiento del algoritmo basado en 2-gram, que se encuentran en la tabla 5.8, se encontró que los mejores resultados de precisión se obtuvieron con el valor de factor de descuento de 0.9. Esto

concuerda con que ese es el valor que fue elegido en [8] y muestra que dar mayor prioridad a las letras iniciales entrega mejores resultados que darles una prioridad más homogénea al resto. Lo anterior, implica que se debería realizar un estudio más profundo para encontrar una solución que entregue mayor prioridad a las primeras letras y, además, considere los efectos de la disminución de la intensidad de líneas en la tabla 5.11, lo que escapa a lo realizado durante la presente memoria.

Por último, comparando los resultados de precisión y velocidad en las distintas configuraciones de podamientos para la Red Convolucional del algoritmo basado en 2-gram, los que se anotaron en la tabla 5.9, se observa que, a medida que más se poda la red, la precisión disminuye y la velocidad aumenta, lo que es consistente con la literatura. Ninguno de los podamientos logró alcanzar el tiempo promedio objetivo de 10 ms por par, aunque la redes podadas lograron una velocidad igual o mejor que la del algoritmo tradicional de Distancia de Coseno de 18 ms por par. En cuanto a los resultados de ranking promedio, se observó que los podamientos de 10 % y de 25 % lograron mantener una precisión levemente mejor al de Levenshtein con *Metaphone*, por lo que son una alternativa viable a la red sin podamiento si es que se necesita priorizar la velocidad sin importar que baje levemente la precisión. Para el podamiento de 35 %, en cambio, se obtuvieron precisiones muy similares al de Levenshtein con *Metaphone*. Finalmente, para los podamientos mayores a 35 %, tanto la velocidad como la precisión, son claramente peores que con ese método tradicional, por lo que no sirven como alternativas a estos y, en particular, el podamiento de 65 % entrega resultados de precisión demasiados bajos para ser utilizado.

Capítulo 7

Conclusiones

En la presenta memoria se hizo un estudio de la literatura necesaria para entender mejor el problema de similitud fonética de marcas y cómo es resuelto por distintos métodos. Además, se planificó y ejecutó la metodología necesaria para extender la base de datos, implementar los distintos algoritmos y comparar los resultados de estos métodos. Dentro de la metodología se compararon 10 configuraciones distintas de algoritmos fonéticos tradicionales, se realizaron modificaciones a una metodología en base de Red Neuronal Recurrente y se implementó el código de un algoritmo con una Red Convolucional desde cero, realizándoles modificaciones para mejorar su velocidad y precisión. También, se realizaron entrenamientos con diferentes conjuntos de entrenamientos y variables para tratar de obtener los mejores resultados posibles.

Se cumplió el objetivo de extender la base de datos que existía previamente, obteniendo así tres bases de datos para entrenar las redes. La primera base fue el resultado de extender manualmente una base de 3.855 pares de marcas fonéticamente similares en inglés a casi el doble de tamaño, obteniendo 6.360 pares de marcas. La segunda base estaba compuesta por 14.250 pares marcas fonéticamente similares en español y se extendió manualmente hasta obtener una base de datos de 15.370 pares. La tercera base consistió en agregar una nueva base de datos de 68.060 pares de alias de nombres que se obtuvo de limpiar manualmente un repositorio encontrado en línea.

También se logró el objetivo de encontrar un método de buscador fonético basado en redes neuronales que obtuviera una mejor precisión que los algoritmos fonéticos tradicionales en los conjunto de datos de prueba creadas a partir de la base de datos extendida. Esto se logró con el algoritmo con Red Neuronal Artificial Convolucional basado en 2-gram entrenado en los conjunto de datos de prueba en inglés más español, utilizando un factor de descuento de 0.9 y sin realizar podamiento. Este algoritmo también logró mantener mejores precisiones que el mejor método tradicional cuando se le podaron un 10 % y un 25 % de los filtros con menor efecto según la norma l1.

En cuanto al objetivo de velocidad, se tiene que no se logró obtener un algoritmo a base de redes neuronales capaz de alcanzar la velocidad objetivo de 30 segundos en analizar la base de datos de INAPI. La red que logró un mejor tiempo de procesamiento es el algoritmo con Red Neuronal Artificial Convolucional basado en 2-gram con un podamiento de 65 %, el cual obtuvo un tiempo de procesamiento cercano a 40 segundos en procesar la base de datos,

pero resultó con una precisión mucho menor que la del algoritmo tradicional. Por lo tanto, si se quiere una solución rápida con mejor precisión que el método tradicional, la red con un podamiento de 25 % logra esto con un tiempo de procesamiento cercana a 50 segundos. Otra alternativa de podamiento un poco más lenta fue el de 10 %, el cual logró una mejor precisión que el de 25 %, con una velocidad cercana a 54 segundos en procesar la base de datos completa; 6 segundos menos que con la red sin podar.

Como contribución de la presente memoria se tiene que se logró abordar un problema práctico, donde, actualmente no existe mucha investigación al respecto. Por lo tanto, se adaptó un algoritmo ya existente que funcionaba para el problema de marcas en coreano e inglés para que pudiera resolver el problema de marcas en inglés y español. Además, se realizaron cambios en su etapa de preprocesamiento para independizarlo del procesamiento, así mejorando la velocidad del método implementado en su uso práctico. Con lo anterior, se obtuvo un buscador fonético a base de redes neuronales que, aunque no alcanzó la velocidad objetivo, estuvo cerca de hacerlo y obtuvo mejores resultados de precisión que los métodos tradicionales vigentes.

Bibliografía

[1] Instituto Nacional de Propiedad Industrial, *DIRECTRICES DE PROCEDIMIENTO DE REGISTRO MARCAS COMERCIALES*. Centro de Documentación de INAPI, pp. 42-404, 2010.

[2] National Archives, *The Soundex Indexing System*. archives.gov 2007.[Online], Available: `http://www.archives.gov/research/census/soundex.html`[Accessed: 4/10/2021].

[3] I. Amón, F. Moreno, and J. Echeverri, *Algoritmo fonético para detección de cadenas de texto duplicadas en el idioma español*. Revista Ingenierías Universidad de Medellín, vol. 11, no 20, pp. 127-138. May 2012.

[4] M. D. P. A. y N. Bailón-Miguel, *Performance of spanish encoding functions during record linkage*. Data Analytics, pp. 9-13. December 2016.

[5] G. Kondrak, *A New Algorithm for the Alignment of Phonetic Sequences*. in 1st Meeting of the North American Chapter of the Association for Computational Linguistics, 2000, pp. 288–295.

[6] K. O'Shea and R. Nas, *An Introduction to Convolutional Neural Networks*. arXiv preprint arXiv:1511.08458, December 2015.

[7] A. Thyagarajan, *Siamese Recurrent Architectures for Learning Sentence Similarity*. in Proceedings of the AAAI conference on artificial intelligence, 2015, Vol. 30, No. 1, pp. 2786–2792.

[8] K. P. Ko, K. H. Lee, M. S. Jang, and G. H. Park, *2-gram-based Phonetic Feature Generation for Convolutional Neural Network in Assessment of Trademark Similarity*. arXiv preprint arXiv:1802.03581, February 2018.

[9] C. Thornton, *Fuzzy Name Matching with Machine Learning*. Towards Data Science, July 2020 [Online], Available: `https://towardsdatascience.com/fuzzy-name-matching-with-machine-learning-f09895dce7b4`[Accessed: 26/07/2021].

[10] F. Patman and L. Shaefer, *Is Soundex Good Enough for You? On the Hidden Risks of Soundex-Based Name Searching*. Language Analysis Systems Inc., Herndon, 2001.

[11] A. Nelson, *Implement Phonetic ("Sounds-like") Name Searches with Double Metaphone Part VI: Other Methods & Additional Resources*. Code Project, March 2007 [Online], Available: `https://www.codeproject.com/Articles/4625/Implement-Phonetic-Sounds-like-Name-Searches-wit-4`[Accessed: 15/10/2021].

[12] K. Koneru and C. Varol, *MetaSoundex Phonetic Matching for English and Spanish.* Global Journal of Enterprise Information System, vol. 10, no 1, pp. 1-13, March 2018.

[13] N. Smetanin, *Phonetic Algorithms.* Blogspot, March 2011 [Online], Available: `https://ntz-develop.blogspot.com/2011/03/phonetic-algorithms.html`[Accessed: 7/12/2021].

[14] Y. Sun, L. Ma, and S. Wang, *A Comparative Evaluation of String Similarity Metrics for Ontology Alignment.* Journal of Information & Computational Science, pp. 957–964. February 2015.

[15] H. Chen, *String Metrics and Word Similarity applied to Information Retrieval.* Master Thesis, University Of Eastern Finland, Finland, 2012.

[16] V. I. Levenshtein, *Binary Codes Capable of Correcting Deletions, Insertions and Reversals.* Soviet physics doklady, Vol. 10, No. 8, pp. 707-710, January 1965.

[17] C. C. Little, *abydos distance package (Cosine).* Abydos, January 2020 [Online], Available: `https://abydos.readthedocs.io/en/latest/abydos.distance.html#abydos.distance.Cosine`[Accessed: 16/10/2021].

[18] T. Mitchell, *Machine Learning.* New York: McGraw Hill, pp. 1-5, 1997.

[19] Lawtomated, *A.I. Technical: Machine vs Deep Learning, April 2019.* [Online], Available: `https://lawtomated.com/a-i-technical-machine-vs-deep-learning/`[Accessed: 29/12/2021].

[20] S. Haykin, *Neural Networks and Learning Machines.* New York: Prentice Hall, pp. 21-40, 3rd ed., 2009.

[21] A. Graves, *Generating Sequences With Recurrent Neural Networks.* arXiv preprint arXiv:1308.0850v5, June 2014.

[22] D. Blalock, J. J. G. Ortiz, J. Frankle, and J. Guttag, *WHAT IS THE STATE OF NEURAL NETWORK PRUNING?* arXiv preprint arXiv:2003.03033, March 2020.

[23] V. Houdebine, *Making Neural Networks Smaller for Better Deployment: Solving the Size Problem of CNNs Using Network Pruning With Keras.* Data Iku, July 2020. [Online] Available: `https://blog.dataiku.com/making-neural-networks-smaller-for-better-deployment-solving-the-size-problem-of-cnns-using-network-pruning-with-keras`[Accessed: 18/10/2021].

[24] J. Ge, *Measuring Short Text Semantic Similarity with Deep Learning Models.* Information Systems and Technology, pp. 1-68, 2018.

[25] M. Mansoor, Z. Rehman, M. Shaheen, and M. Khan, *Deep Learning based Semantic Similarity Detection using Text Data.* Information Technology and Control, 2020, vol. 49, no 4, pp. 495-510.

[26] M. S. Walia, *Measuring Text Similarity Using BERT, May 2019.* Analytics Vidhya [Online], Available: `https://www.analyticsvidhya.com/blog/2021/05/measuring-text-similarity-using-bert/`[Accessed: 2/1/2021].

[27] Y. Shao, *HCTI at SemEval-2017 Task 1: Use convolutional neural network to evaluate Semantic Textual Similarity.* in Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017), 2017. pp. 130-133.

[28] Instituto Mexicano de la Propiedad Industrial, *Búsqueda fonética.* gob.mx, 2016 [Online], Available: `https://acervomarcas.impi.gob.mx:8181/marcanet/vistas/common/datos/bsqFoneticaCompleta.pgi`[Accessed: 1/4/2021].

[29] Organización Mundial de la Propiedad Intelectual, *WIPO Global Brand Database.* wipo.int, Madrid Online Services. [Online] Available: `https://www3.wipo.int/branddb/en/#`[Accessed: 1/6/2021].

[30] C. Thornton, *HMNI, September 2020.* Repositorio Público de Github [Online], Available: `https://github.com/Christopher-Thornton/hmni`[Accessed: 30/8/2021].

[31] O. Tursuna, C. Akera, and S. Kalkan, *A Large-scale Dataset and Benchmark for Similar Trademark Retrieval.* arXiv preprint arXiv:1701.05766, October 2017.

[32] Instituto Nacional de Propiedad Industrial, *MARCAS Datos Abiertos, 2017.* Repositorio de Marcas Registradas y Solicitadas [Online], Available: `https://ion.inapi.cl/Marca/Opendata.aspx`[Accessed: 15/1/2021].

[33] T. Ogawa, *num2words library - Convert numbers to words in multiple languages.* Repositorio Público de Github, June 2021 [Online], Available: `https://github.com/savoirfairelinux/num2words`[Accessed: 16/2/2021].

[34] A. Haapala, *Python-Levenshtein, February 2021.* Repositorio Público de Github [Online], Available: `https://github.com/ztane/python-Levenshtein/`[Accessed: 29/9/2021].

[35] L. Prechelt, *An empirical comparison of C, C++, Java, Perl, Python, Rexx, and Tcl for a search/string-processing program.* Universitat Karlsruhe, Karlsruhe, Germany, pp. 28-30, March 2000.

[36] C. C. Little, *Abydos, December 2020.* Repositorio Público de Github [Online], Available: `https://github.com/chrislit/abydos`[Accessed: 20/1/2021].

[37] C. C. Little, *abydos distance package (Levenshtein).* Abydos, 2020 [Online], Available: `https://abydos.readthedocs.io/en/latest/abydos.distance.html#abydos.distance.Cosine`[Accessed: 16/10/2021].

[38] Mphilli, Mitchellpkt, CanadianCommander, and Timvancann, *English to IPA (eng_to_ipa), April 2020.* pypi.org [Online], Available: `https://pypi.org/project/eng-to-ipa/`[Accessed: 5/2/2021].

[39] Google Inc., *TensorFlow.* Repositorio Público de Github, June 2021 [Online], Available: `https://github.com/tensorflow/tensorflow`[Accessed: 14/2/2021].

[40] Google Inc., *Welcome to Colab!* Introducción a la nube de Colab [Online], Available: `https://colab.research.google.com`[Accessed: 24/12/2021].

[41] B. Whetton, *Keras-surgeon, April 2021.* Repositorio Público de Github [Online], Available: `https://github.com/BenWhetton/keras-surgeon`[Accessed: 20/10/2021].

ANEXOS

Anexo A

Estudio completo sobre la Base de Datos de INAPI

Estudio sobre la Base de Datos de 292706 marcas registradas de INAPI, donde se obtuvieron los distintos caracteres especiales que no corresponden al alfabeto latino, separada en las tres tablas A.1, A.2 y A.3 debido a la cantidad de filas.

A.1. Estudio Sobre la Base de Datos de INAPI Parte 1

Tabla A.1: Lista completa de caracteres especiales encontrados en la Base de Datos de 292706 marcas registradas en la INAPI y sus respectivos porcentajes. Parte 1

Carácter Especial	Porcentaje en la Base de Datos de INAPI
.	43.14 %
-	30.12 %
,	16.55 %
&	12.48 %
’	10.00 %
!	3.90 %
+	2.76 %
”	1.68 %
´	1.01 %
:	0.96 %
/	0.93 %
)	0.89 %
(	0.81 %
’	0.56 %
@	0.54 %
%	0.52 %
?	0.47 %
¡	0.39 %
°	0.38 %
Ö	0.37 %
\n	0.35 %
#	0.35 %
·	0.33 %
•	0.26 %
º	0.25 %
Ä	0.25 %
_	0.21 %
Ç	0.17 %
*	0.16 %
–	0.15 %
‘	0.14 %

A.2. Estudio Sobre la Base de Datos de INAPI Parte 2

Tabla A.2: Lista completa de caracteres especiales encontrados en la Base de Datos de 292706 marcas registradas en la INAPI y sus respectivos porcentajes. Parte 2

Carácter Especial	Porcentaje en la Base de Datos de INAPI
¿	0.14 %
$	0.14 %
Ê	0.12 %
È	0.11 %
Ë	0.10 %
;	0.10 %
ö	0.10 %
ä	0.08 %
Ô	0.08 %
Ã	0.07 %
>	0.07 %
è	0.07 %
ê	0.07 %
=	0.06 %
Ï	0.06 %
]	0.06 %
[	0.06 %
|	0.05 %
À	0.04 %
²	0.03 %
Â	0.03 %
à	0.03 %
<	0.02 %
Ò	0.02 %
...	0.02 %
{	0.02 %
ô	0.02 %
Ø	0.02 %
ë	0.02 %
Ù	0.02 %

A.3. Estudio Sobre la Base de Datos de INAPI Parte 3

Tabla A.3: Lista completa de caracteres especiales encontrados en la Base de Datos de 292706 marcas registradas en la INAPI y sus respectivos porcentajes. Parte 3

Carácter Especial	Porcentaje en la Base de Datos de INAPI
“	0.02 %
ù	0.01 %
ø	0.01 %
ã	0.01 %
}	0.01 %
‘	0.01 %
Õ	0.01 %
Û	0.01 %
Å	0.01 %
å	0.01 %
Ì	0.01 %
˜	0.01 %
â	0.01 %
ç	0.01 %
a	0.01 %
\	0.01 %
Ý	0.01 %
¨	0.01 %
Î	0.01 %
õ	0.00 %
§	0.00 %
¯	0.00 %
\t	0.00 %
ì	0.00 %
ˆ	0.00 %
®	0.00 %
ß	0.00 %
ò	0.00 %
”	0.02 %
ȝ	0.01 %

Anexo B

Intensidad de líneas

Los valores de intensidad de líneas obtenida al variar el valor de factor de descuento se presentan en la tabla B.1

Tabla B.1: Lista completa de valores de la matriz de Intensidad de líneas para los 3 factores de descuento distintos elegidos.

Factor de Descuento	Valores de Intensidad de Líneas
0.90	[255, 229, 185, 135, 88, 52, 27, 13, 5, 2, 0, ...]
0.95	[255, 242, 218, 187, 152, 118, 86, 60, 40, 25, 15, 8, 4, 2, 1, 0....]
0.99	[255, 252, 247, 240, 230, 219, 206, 192, 177, 162, 146, 131, 116, 102, 88, 76, 65, 54, 45, 37, 30, 25, 20, 15, 12, 9, 7, 5, 4, 3, 2, 1, 1, 0....]