Universidad Autónoma de Querétaro

Facultad de Ingeniería

Doctorado en Ingeniería

CLASIFICACIÓN SUPERVISADA DE SERIES DE TIEMPO MEDIANTE SU CARACTERIZACIÓN TEMPORAL A TRAVÉS DE MÉTODOS ACTUARIALES.

TESIS

Que como parte de los requisitos para obtener el grado de

Doctor en Ingeniería

Presentan:

Wilfrido Jacobo Paredes Garcia

Dirigido por:

Dr. Roberto Augusto Gómez Loenzo

SINODALES

Dr. Roberto Augusto Gómez Loenzo Presidente	Firma
Dr. Juvenal Rodríguez Resendiz Secretario	Firma
Dr. Eduardo Castaño Tostado Vocal	Firma
Dr. Arturo González Gutíerrez Suplente	Firma
Dra. Adriana Medellín Suplente	Firma

Dr. Manuel Toledano Ayala Director de la Facultad	Dra. Ma. Guadalupe Flavia Loarca Piña Director de Investigación y Postgrado

Centro Universitario
Querétaro, Qro.
Diciembre de 2020
México

Resumen

El propósito de esta tesis es determinar si son o no significativos los metadatos usados en la recomendación musical, puesto que esto limita la experiencia musical a géneros y artistas muy similares.

En este trabajo se presenta una propuesta de clasificación de canciones mediante el análisis matemático y estadístico de sus elementos temporales, es decir características propias de la música, generando así una alternativa a algunos clasificadores actuales que tienden a incluir en la caracterización de las canciones elementos ajenos a la música en sí, tales como año, artista, calificación, popularidad entre otras. Para ello se optó por utilizar el método de clasificación *K Nearest Neighbors*.

Palabras clave: Recomendación musical, Aprendizaje supervisado, K Nearest Neighbors, Clasificación

Abstract

The aim of this thesis is to determine whether or not the usage of metadata in musical recommendation is significant, since this restricts the musical experience to very similar artists and genres.

This thesis presents a proposal for music classification through the mathematical and statistical analysis of the temporal elements of the songs, creating an alternative to some current classifiers that include other elements, such as year, artist, rating, among others. In order to create this classifier, the algorithm K-Nearest Neighbors (KNN) was used.

Key words: Music recommendation, supervised learning, K Nearest Neighbors, Classification

Esta tesis está dedicada a nuestras familias, epsecialmente a nuestros padres, quienes siempre nos han mostrado su apoyo, nos brindan su amor y nos motivan día con día. A nuestros amigos, que nos acompañaron a lo largo de este proceso y han estado en las buenas y en las malas.

Agradecimientos

Quisiera agradecer a mi padres que me apoyaron en estos 4 años de trabajo, al Dr. Roberto Augusto Gómez por su apoyo y comprensión durante este trayecto que fue el doctorado; al Dr. Juvenal Rodríguez por todo su apoyo al permitirme involuvrarme en otros proyectos que me permitieron tener un panorama más amplío de la investigación; al Dr. Eduardo Castaño por su apoyo y total disposición en estos años de trabajo; al Dr. Arturo González por su excelente tu tutoria y atención en en estos años y a la Dra. Adriana Medellín por su asesoría brindada en la realización de este trabajo.

Una mención especial a mi amigos y profesores, que por mucho o poco que hayamos interactuado me ayudaron a formar parte de la persona que logró presentar este trabajo, y que, sin ellos existió la posibilidad de que no se redactaran estos pequeños párrafos que vendrán a continución.

Finalmente un agradecimiento total a la Universidad Autónoma de Querétaro y al Consejo Nacional de Ciencia y Tecnología (CONACyT) por la oportunidad y apoyo que me brindó bajo la beca

Prefacio

Quisiera comentar que esta obra inició con un gran objetivo más allá de lo plasmado en el protocolo de investigación que redacté hace ya cuatro años. He de ser sincero, el gran objetivo, que era una especie de sueño vívido no se cumplió porque en éste, mi primer trabajo autónomo de investigación, el camino no resultó como había sido trazado. Sin embargo, espero que el camino recorrido y los logros alcanzados en este trabajo permitan que futuros interesados tengan un recorrido más llevadero, al menos, en sus inicios.

Sobre esta obra, primero debo comentar que he decido eliminar el regular apéndice de código fuente. No creo, sinceramente que haga falta. En su lugar, el código fuente empleado, así como las bases de datos y documentos fuentes pueden ser encontrado en mi GitHub particular para su libre uso.

`https://github.com/Wilfridovich17/doctorado`

Inicialmente, este trabajo fue realizado usando una mezcla de R y Python. R para algunas tareas y Python para otras. Parte final de la realización de este trabajo fue trasladar todo el código de R a Python. Resultó molesto, durante la elaboración de este trabajo, alternar entre ambos ambientes e, imagino que si estás leyendo esto, también lo sería para ti. Por ello, todo el código fuente que encontrarás en mi GitHub está escrito en Python 3.8.

Por otro lado, si la fortuna me sonríe y la vida lo permite, espero poder seguir dando mantenimiento al repositorio con el fin de tener un equipo y proyecto de trabajo que de prioridad al español, ya que al 2020 el único método de fácil uso para realizar todas las tareas escritas para el caso del español, es el proyecto de la Universidad de Stanford.

Wilfrido J. Paredes

Santiago de Querétaro, Qro. 12 de marzo de 2021

Índice general

Índice de tablas

Índice de figuras

CAPÍTULO 1

Introducción

CAPÍTULO 2

Marco teórico

Al hablar de PLN es necesario hablar sobre teoría del lenguaje y como modelar esa teoría de forma que sea práctica llevarla a la parte computacional. De esta forma, este capítulo trata de ser un compendio de la teoría del lenguaje y cómo fue llevada a la practica computacional.

La Fig 2.1 muestra la estructura del capítulo. En particular, la parte medular de este apartado está concentrado en las primera dos secciones: Modelos del lenguaje y Modelos computacionales con base en la hipótesis distributiva. Adicional a esto, existen cuatro secciones: una de ellas enfocadas a cómo procesar la información para construir las entradas del modelo, Pre-procesamiento; otra enfocada a como se optimiza el proceso de estimación de los modelos empleados, Optimización; una tercera sección que discute los elementos abordados en la propuesta de trabajo, Notas Adicionales sobre el Procesamiento de Lenguaje Natural y; finalmente una sección con el objetivo de funcionar como una especie de glosario sobre algunos conceptos lingüísticos que sería poco práctico desglosar en los pies de página.

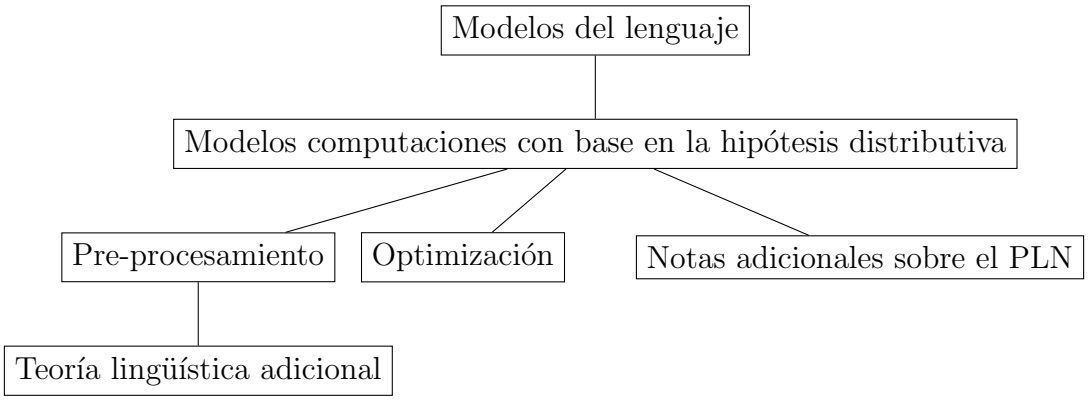


Figura 2.1: Organización del capítulo.

2.1 Modelos del lenguaje

Antes de hablar sobre modelos de lenguaje resulta necesario mencionar que existen dos enfoques, el enfoque generativo de Chomsky[1] y el enfoque distributivo de Harris[2]. Aunque, el enfoque de Harris es la base del PLN actual, es imposible hablar de modelos de lenguaje natural dejando de lado en enfoque generativo de Chomsky.

[1]El enfoque de Noam Chomsky está plasmado en su trabajo doctoral "Transfrormational Analysis" y en el libro derivado "The logical structure of linguistic theory" escrito en 1955.

[2]La base del enfoque de Zellig Harris fue publicado en su trabajo "Distributional structure" publicado en 1954.

2.1.1 Gramática Generativa de Chomsky

¿Por qué no debe usarse un modelo estadístico?

Las gramática generativa de Chomsky parte de un análisis sintáctico de la lengua con el objetivo fundamental de encontrar un "gramática" que permita discriminar entre oraciones gramáticales y oraciones agramáticales, es decir, separar las oraciones que pertenecen a la lengua de las que no pertenecen. En ese sentido, se puede entender a la grámatica de Chomsky como un modelo, un modelo que permite clasificar oraciones en válidas o no válidas.

Dada la naturaleza desconocida de la gramática, el punto de partida del modelo de Chomsky radica en cuáles no deben ser las bases del modelo, en particular, el modelo o gramática no puede ser estadístico. Chomsky (1974) justifica lo anterior en tres puntos:

1. Cualquier colección de locuciones solo es una proyección de la gramática. En ese sentido, la gramática no debe partir de una colección de locuciones sino que debe ser capaz de recrear dicha colección.

2. En el terreno de la estadística se habla sobre si un modelo es significativo o no. Pero el calificativo de gramatical no es semánticamente equivalente al significado de significativo. El adjetivo tiene significativo tiene que ver con la coherencia de los datos con el modelo, pero la gramáticalidad no, una oración puede ser gramátical y absurda a la vez.

3. Es imposible hablar de tener una muestra con un alto orden de aproximación a la lengua, pues el conjunto de locuciones gramaticales que se pueden formar en la lengua es infinito.

Por lo tanto, el enfoque de Chomsky parte de que no basta con solo recolectar locuciones.[3]

Gramáticas de Contexto Libre

La Grámatica de Contexto Libre (GCL) es uno de los tres componentes de la Gramática Generativa de Chomsky (en particular, Chomsky denomina a esta parte como la *Estructura Sintáctica* de su gramática). Cohen (1991) define a una GCL como una colección de tres elementos: un alfabeto de terminales denotado por Σ, un conjunto de símbolos no terminales denotado por S y un conjunto finito de producciones[4] de la forma:

$$\text{Elemento no terminal} \rightarrow \text{Cadena finita de terminales y/o no terminales}$$

Usando esta estructura de producciones es posible generar ciertas oraciones, donde en el caso particular de Chomsky, los elementos no terminales en una primera instancia

[3] De acuerdo con Orduña López (2011), una locución puede definirse en un sentido estricto y en un sentido amplío. En un sentido estricto, son lexemas con estructura fraseológica y pueden clasificarse en sustantivos, pronombres, adjetivos, determinantes, verbos, adverbios, preposiciones y conjunciones. En otras palabras una locución es una palabra. En un sentido amplío una locución es un elemento oracional, los cuales pueden pueden clasificarse en verbales, adjetivales, adverbiales, preposicionales y conjuntivas. Así frases como "el comal le dijo a la olla" son una locución en sentido amplío.

[4] Chomsky nombra a las producciones como "ahormaciones", aunque otros autores también les llaman derivaciones.

pueden ser constituyentes y los elementos terminales resultan ser morfemas[5]. Chomsky (1974) usa el ejemplo la oración "the woman hit the ball". Para ello, propone una serie de producciones como las siguientes:

$$\begin{aligned} S &\rightarrow FN + FV \\ FN &\rightarrow DET + N \\ FV &\rightarrow VERBO + FN \\ DET &\rightarrow \text{the} \\ N &\rightarrow \text{ball, woman} \\ VERBO &\rightarrow \text{hit} \end{aligned}$$

Sin embargo, es sencillo adecuar esto para el caso del español. Considere las siguientes producciones:

$$\begin{aligned} S &\rightarrow FN + FV \\ FN &\rightarrow DET + N, NP \\ FV &\rightarrow VERBO + FN \\ DET &\rightarrow \text{un} \\ N &\rightarrow \text{libro} \\ NP &\rightarrow \text{Sarahí} \\ VERBO &\rightarrow \text{compró} \end{aligned}$$

Usando estas producciones resulta posible generar oraciones como "Sarahí compró un libro", "Un libro compró Sarahí" y absurdas como "Un libro compró un libro" o "Sarahí compró Sarahí". Pero la adecuación no resulta compleja entre ambos idiomas.

Finalmente, algo que es necesario resaltar, es que, en teoría de la computación un lenguaje no es más que un conjunto de palabras y el alfabeto, Σ, son los caracteres empleados para formar a las palabras. Sin embargo, Chomsky no habla de palabras sino de oraciones, en su caso, el lenguaje es un conjunto de oraciones (las oraciones gramáticales) y el alfabeto de ese lenguaje los morfemas en una primera instancia.

Transformaciones

El segundo elemento de la Gramática Generativa de Chomsky son las transformaciones. Chomsky (1974) menciona que las transformaciones son como un puente entre la estructura sintáctica y la estructura morfofonémica, en el sentido que la estructura sintáctica produce oraciones con elementos terminales pero que pueden que no estén en el orden correcto, hagan faltan morfemas o sobren morfemas. En ese sentido, las transformaciones son reglas que permiten permutar subcadenas de la oración, eliminar elementos terminales o agregar elementos terminales.

Para hablar de transformaciones, Chomsky (1956) primeramente define el siguiente concepto:

Definición 1 *Un marcador de frase K de una oración S es un conjunto de cadenas que*

[5]De acuerdo con Allerton (1979) los morfemas también son constituyentes, pero son constituyentes finales, lo que resulta en una explicación de porque Chomsky los elige como elementos terminales de su estructura sintáctica. Por simplicidad uno puede pensar en los morfemas como palabras.

ocurren resultado de una serie de producciones $P_1, P_2, \ldots, P_n$. *Además se dice que el par ordenado* (S, K) *es analizable en* $(X_1, X_2 \ldots, X_n)$ *si y solo si, existen cadenas* $s_1, s_2 \ldots, s_n$ *tales que* $S = s_1 + s_2 + \cdots s_n$ *y para cada* $i \leq n$

define dos tipos de transformaciones: elemental y derivada, la primera obligatoria y la segunda opcional.

De acuerdo con Allerton (1979) las oraciones obtenidas por la estructura sintáctica son pre-oraciones y al aplicar una transformación elemental se transforman en oraciones, oraciones que se consideran nucleares[6] y, mientras que una

$$
\begin{aligned}
S &\rightarrow FN + FV \\
FN &\rightarrow ART + N \\
FV &\rightarrow VERBO + FN \\
ART &\rightarrow \text{el, la} \\
N &\rightarrow \text{arból, fruta} \\
VERBO &\rightarrow AUX + V \\
V &\rightarrow \text{tiene} \\
AUX &\rightarrow T + M \\
M &\rightarrow ha, ha + estar - participio, estar - preseteT \quad \rightarrow pasado, presente, continuo
\end{aligned}
$$

Morfofonémica

El tercer elemento de la Gramática Generativa de Chomsky es la *Estructura Morfofonémica*. Chomsky (1956) explica esto con la siguiente oración "the man had been taking the book". Para generar tal oración es necesario el siguiente conjunto de producciones:

$$
\begin{aligned}
S &\rightarrow FN + FV \\
FN &\rightarrow DET + N \\
FV &\rightarrow VERBO + FN \\
DET &\rightarrow \text{the} \\
N &\rightarrow \text{man, book} \\
VERBO &\rightarrow AUX + V \\
V &\rightarrow \text{take} \\
AUX &\rightarrow C(M)(haveen)(being) \\
M &\rightarrow \text{will, can, shall, may, must} \\
C &\rightarrow \text{past, present}
\end{aligned}
$$

una transformación:

$$AF\ V \rightarrow V\ AF$$

[6]Chomsky las llama oraciones hormacionales pues son resultado de aplicar primordialmente la estructura sintáctica.

y un conjunto de reglas morfofonémicas que conviertan morfemas en fonemas, las cuáles para Chomsky conforman la estructura morfofonémica de su gramática generativa.

$$
\begin{aligned}
\textit{have past} &\rightarrow \textit{had} \\
\textit{be en} &\rightarrow \textit{been} \\
\textit{take ing} &\rightarrow \textit{taking} \\
\textit{will past} &\rightarrow \textit{would} \\
\textit{can past} &\rightarrow \textit{could} \\
M\ \textit{present} &\rightarrow M \\
\textit{take past} &\rightarrow \textit{took}
\end{aligned}
$$

De esta forma, siguiendo las producciones es posible llegar a "the man past have en be ing take the book", luego al aplicar la transformación se obtiene "the man past have be en take ing the book" y finalmente aplicando la estructura morfofonémica se obtiene "the man had been taking the book". De esta forma es posible generar un conjunto más grande de oraciones que con solo el uso de la estructura sintáctica. Sin embargo, note que a diferencia de la GCL una adecuación directa al español no es posible.

2.1.2 Estructura distributiva de Harris

Chomsky (1977) menciona que la gramática generativa resulta adecuada para los primeros niveles de descripción lingüística: el nivel fonológico y el nivel sintáctico. Sin embargo, no es resulta ser adecuada para un nivel semántico. En ese sentido, el trabajo de Harris, aunque en cierta medida contrapuesto a los objetivos de Chomsky, propone una solución.

Harris separa su trabajo de Chomsky, en el sentido que Chomsky busca un modelo, un dispositivo que permite la adquisición de lenguaje por parte de los seres humanos. En ese sentido las producciones, transformaciones y las reglas morfofonémicas son elementos que conforman ese dispositivo innato en las personas. Por el contrario, Harris (1954) menciona que el lenguaje posee una estructura, aunque no está del todo claro que esa estructura esté presente en los hablantes. Además tal estructura puede dar luz sobre el significado, es decir, encontrar patrones que puedan usarse para la descripción lingüística en un nivel semántico.

En contraste con lo propuesto por Chomsky, el trabajo de Harris, no proporciona un modelo como tal, sino una metodología conocida como metodología distributiva (Sahlgren, 2008). La metodología distributiva se plantea como una forma de explorar lo que hoy se conoce como "Hipótesis distributiva", la cual afirma lo siguiente:

> *Se puede establecer la similaridad semántica entre dos palabras si se considera que para tres palabras o morfemas A, B y C. A difiere más en significado de B que de C, si la distribución de A y B difiere más que la distribución de A y C* (Harris, 1954).

Para dar un ejemplo de lo anterior Harris menciona que, de ser cierta la hipótesis, dado que el significado de *oculista* difiere más del significado de *abogado* que del significado de *médico*, entonces sus distribuciones deberían mantener dicha relación[7]. En ese

[7]Harris no menciona una forma de medir la diferencia entre distribuciones.

sentido, bajo la hipótesis distributiva, debería haber una diferencia más marcada entre la distribución de oculista y abogado que respecto de la distribución de oculista y médico. La Fig. 2.2 se muestra la distribución de la palabra oculista y se muestra como existe una concordancia mayor entre términos oculista y médico que entre los términos oculista y abogado. En ese sentido, la figura ilustra hipótesis distributiva.

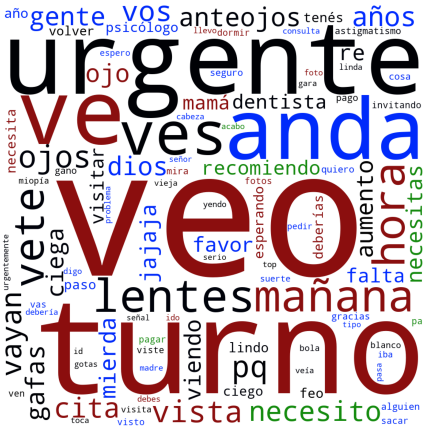


Figura 2.2: Distribución de la palabra oculista. En rojo se muestran los términos concurrentes únicamente con la palabra médico, de verde los términos concurrentes exclusivamente con la palabra abogado y en azul los términos concurrentes con ambas palabras. Gráfico de elaboración propia.

2.2 Modelos computacionales con base en la hipótesis distributiva

Otero (1975) menciona que muchos autores definen la conducta lingüística como "el uso de la palabras", aunque desde su perspectiva debería limitarse a ser definida como "el uso de la palabra" porque se pueden usar palabras sin transmitir un mensaje. ¿Cómo saber que se tiene el uso de la palabra? Desde la perspectiva del PLN esto conoce como Modelado del Lenguaje (ML) y se centra en dos tareas específicas: dada una secuencia de palabras determinar la palabra siguiente y dada una palabra determinar su estructura.

2.2.1 N-gramas

Los modelos que emplearon N-gramas fueron un primer intento de representar las palabras y de resolver la tarea del ML. La idea consiste en calcular la probabilidad de ocurrencia de una secuencia de palabras de longitud L. De acuerdo con Brants et al. (2007) dada una secuencia de palabras de longitud L, $(w_{i_1}, w_{i_2}, \ldots, w_{i_L})$, su probabilidad de ocurrencia puede calcularse como:

$$P\left[W_1 = w_{i_1}, \ldots, W_L = w_{i_L}\right] = P\left[W_1 = w_{i_1}\right] \prod_{j=1}^{L-1} P\left[W_{j+1} = w_{i_{j+1}} | W_1 = w_{i_1}, \ldots, W_j = w_{i_j}\right] \quad (2.1)$$

donde w_j denota la i-ésima palabra del vocabulario, $\{i_j\}_{j=1}^{L}$ es una colección de índice tales que $i_j = 1, 2, \cdots, V$ con V el total de palabras que conforman el vocabulario. De esta forma, las palabras se codifican usando un número natural, sin una preferencia de asignación. De esta forma, la mejor predicción para la palabra siguiente, w_{L+1}, dada una secuencia de longitud L es aquella tal que:

$$\hat{W}_{L+1} = \arg \max_{1 \leq j \leq V} P\left[W_1 = w_{i_1}, \ldots, W_L = w_{i_L}, W_{L+1} = w_j\right] \tag{2.2}$$

Note que (2.1) es un conocido resultado de probabilidad[8] y que implica genera una cadena con base en las probabilidades condicionadas, es decir, la probabilidad de que la segunda palabra sea cierta palabra depende de la palabra que le antecede, la probabilidad de que la tercera palabra sea una palabra en particular depende de las dos palabras que le anteceden y, así sucesivamente (Fig. 2.3).

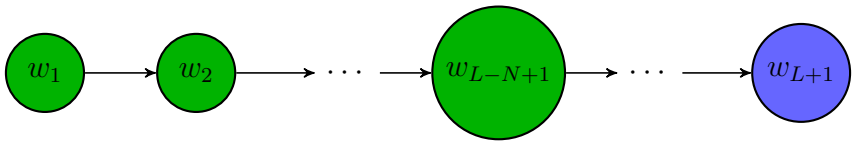


Figura 2.3: Diagrama de una cadena de longitud $L+1$. Donde se muestra que la palabra w_{L+1} depende de todas las palabras anteriores. Gráfico de elaboración propia.

El cómputo (2.1) se vuelve muy costoso desde el punto de vista de la información requerida L es grande. Norvig (2009) afirma que el cómputo completo para $L = 5$ necesitaría aproximadamente 30 GB de información y, es claro que una secuencia de solo 5 palabras es muy limitada. En ese sentido, un supuesto razonable planteado por Brants y su equipo es que la probabilidad de la palabra $L + 1$ solo depende de las N palabras anteriores a ésta y no de toda las palabras anteriores (Fig. 2.4). De esta forma se tiene la siguiente expresión que se considera el modelo de n-gramas:

$$P\left[W_1 = w_{i_1}, \ldots, W_L = w_{i_L}\right] = P\left[W_1 = w_{i_1}\right] \prod_{j=1}^{L-1} P\left[W_{j+1} = w_{i_{j+1}} | W_{j-N+1} = w_{i_{j-N+1}}, \ldots, W_j = w_{i_j}\right] \tag{2.3}$$

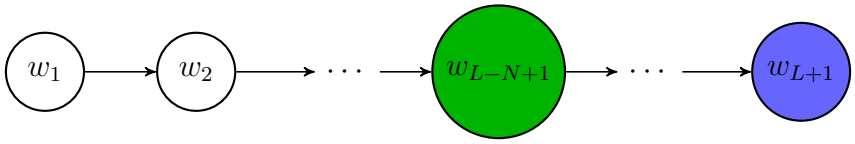


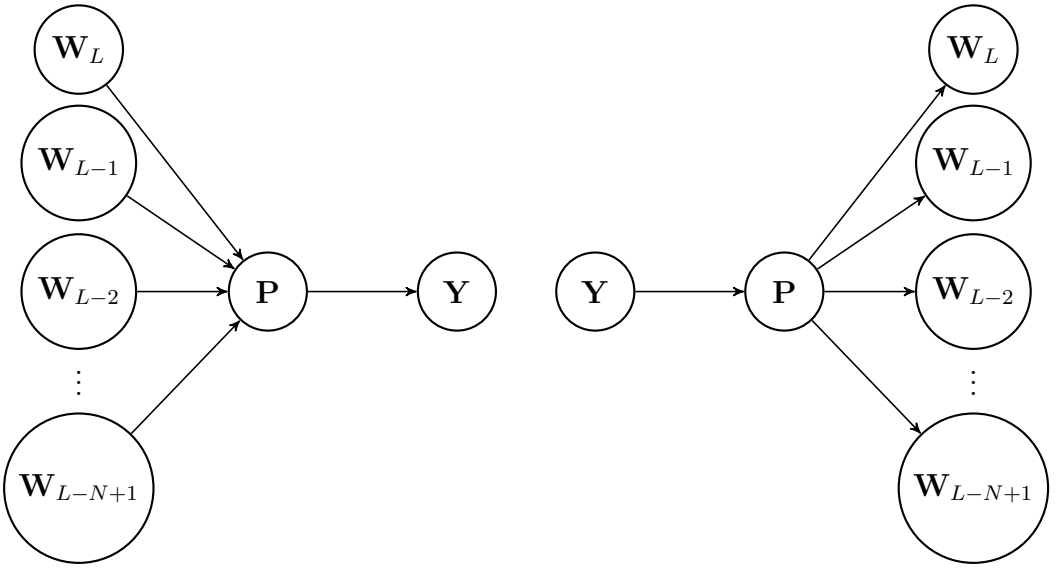
Figura 2.4: Diagrama de una cadena de longitud $L+1$. Donde se muestra que la palabra w_{L+1} depende solo de un número limitado de palabras (en verde). Gráfico de elaboración propia.

2.2.2 Modelos CBOW y Skip-Gram

Desde el punto de vista de los modelos basados en n-gramas emplean la distribución de las palabras para cuantificar la incertidumbre sobre la siguiente palabra; pero no permite saltar al nivel semántico. Además de que puede ser cuestionable la reducción

[8] Se encuentra como ejercicio en el primer capítulo de Introduction to Probability Models de Sheldon Ross.

para hacer computacionalmente el modelo. Mikolov et al. (2013a,b) proponen que para alcanzar el nivel semántico es necesario emplear otros enfoques. En su trabajo el propone dos modelos: el Modelo de Bolsas de Palabras Continuo (CBOW por sus siglas en inglés) y el modelo Skip-Gram. Ambos modelos basados en redes neuronales secuenciales con tres capas: una capa de entrada, una capa de proyección y una capa de salida (Fig. 2.5).



(a) Diagrama de la arquitectura del modelo CBOW. Gráfico de elaboración propia.

(b) Diagrama de la arquitectura del modelo Skip-Gram.

Figura 2.5: Modelos propuestos por Mikolov para representación y procesamiento de palabras.

El modelo CBOW está enfocado en resolver la misma tarea que los modelos basados en n-gramas: predecir la palabra siguiente dada una sucesión de palabras. Por otro lado, el modelo Skip-Gram permite determinar la distribución de las palabras adyacentes a una palabra dada. En ese sentido, se puede entender que el modelo Skip-Gram es el modelo *inverso* del modelo CBOW. Sin embargo, el enfoque permite:

1. Manejar una cantidad más amplía de palabras contexto. Por ejemplo, su trabajo se emplearon 10 palabras contexto.

2. La proyección permite de alguna manera rescatar la estructura semántica que menciona Harris.

3. La estructura permite inferir el significado de palabras desconocidas.

El modelo propuesto por Mikolov usualmente se le conoce como *word2vec* y es el enfoque se sigue empleando a la fecha con algunas modificaciones, pero sin perder su esencia. ¿Pero como funciona el modelo? Para ello resulta necesario definir algunos términos. El primer término es la codificación One-Hot, la cual, de acuerdo con Bruce et al. (2020) se emplean en modelos

Definición 2 (Codificación One-Hot) *Dada una variable categórica X con p niveles, una observación de ésta puede codificarse como un vector binario X' de dimensión p donde* 1 *índica el nivel observado.*

En los modelos basados en N-gramas cada palabra se puede considerar una variable aleatoria de V niveles, donde V es el número de palabras del vocabulario como ya se mencionó en el apartado anterior. Entonces si la primera palabra de un texto W_1 es la i-ésima palabra del vocabulario, W_1 puede codificarse como $\left[\underbrace{0,\ldots 0}_{i-1},1,\underbrace{0,\ldots,0}_{V-i}\right]'$.

Luego, según Rong (2014), el modelo planteado por la Fig. 2.5a supone que cada palabra se codifica usando One-Hot. En ese sentido, el empleo de N palabras contexto implicaría el empleo de $N \times V$ neuronas de entrada. Sin embargo, un supuesto del modelo de Mikolov consiste en considerar que está proyección es invariante a la posición, es decir, que la información que proporciona una palabra al contexto de otra no depende de su distancia a la palabra sino simplemente si está dentro de una vecindad definida *a priori*[9]. En ese sentido, dada una secuencia de L palabras si el contexto de la palabra $L+1$ solo depende de las N palabras anteriores, se tiene que si $\mathbf{P}$ un vector p dimensional es el vector proyección de la capa oculta, entonces:

$$\mathbf{P} = \mathbf{\Omega}_1' \mathbf{W}$$

donde $\mathbf{W} = \frac{1}{N}\sum_{i=L-N+1}^{L} \mathbf{W}_i$ es un vector de dimensión V y $\mathbf{\Omega}_1$ es la matriz de pesos asociada a la transición entre la capa de entrada y la capa de proyección. Finalmente, lo que se calcula un vector de probabilidades $\mathbf{Y}$ de dimensión V donde cada entrada está asociada a cada una de las palabras del vocabulario. Para ello se emplea la función de activación *softmax*. Así, si $\mathbf{U}$ es el vector proyección resultado de proyectar $\mathbf{P}$, es decir:

$$\mathbf{U} = \mathbf{\Omega}_2' \mathbf{P}$$

Entonces, la j-ésima entrada de $\mathbf{Y}$ se puede calcular como:

$$y_j = \frac{e^{u_j}}{\sum_{k=1}^{V} e^{u_k}}$$

la anterior expresión corresponde a la función de activación *softmax* y u_j corresponde a la j-ésima entrada el vector $\mathbf{U}$. Similarmente, el modelo planteado por la Fig. 2.5b puede entenderse de manera análoga, solo que en este caso

$$\mathbf{P} = \mathbf{\Omega}_1' \mathbf{W}$$

con $\mathbf{W}$ donde es la representación One-Hot de la palabra a proyectar. Note que no se cambia la matriz de pesos del modelo CBOW, Goldberg y Levy (2014) mencionan que la matriz de pesos para el modelo Skip-Gram es la misma que para el modelo CBOW. Más aún, debido a la codificación de la palabra a proyectar, la proyección de ésta consiste únicamente de una de las columnas de $\mathbf{\Omega}'$. Por lo tanto, $\mathbf{\Omega}$ posee la información semántica de todo el vocabulario.

Bellegarda y Monz (2016) mencionan que $\mathbf{\Omega}$ se le conoce como *embebido de palabras*[10]. Además, afirman que es en el espacio vectorial formado por todas las posibles proyecciones donde se encuentra representadas la estructura semántica (Fig. 2.6) y donde

[9] De ahí el nombre de bolsa de palabras, es como si las palabras se metieran en una bolsa y es esa bolsa la que da el contexto de la palabra.

[10] También puede usar el concepto "encaje de palabras".

es posible realizar operaciones como

$$\mathbf{e}_{\text{rey}} - \mathbf{e}_{\text{hombre}} + \mathbf{e}_{\text{mujer}} = \mathbf{e}_{\text{reina}}$$

donde $\mathbf{e}_{w_j}$ es la proyección de la palabra w_j. Sin embargo, Goldberg y Levy (2014) menciona que no se conoce la razón por la que el enfoque funciona y que lo único claro es que funciona[11]

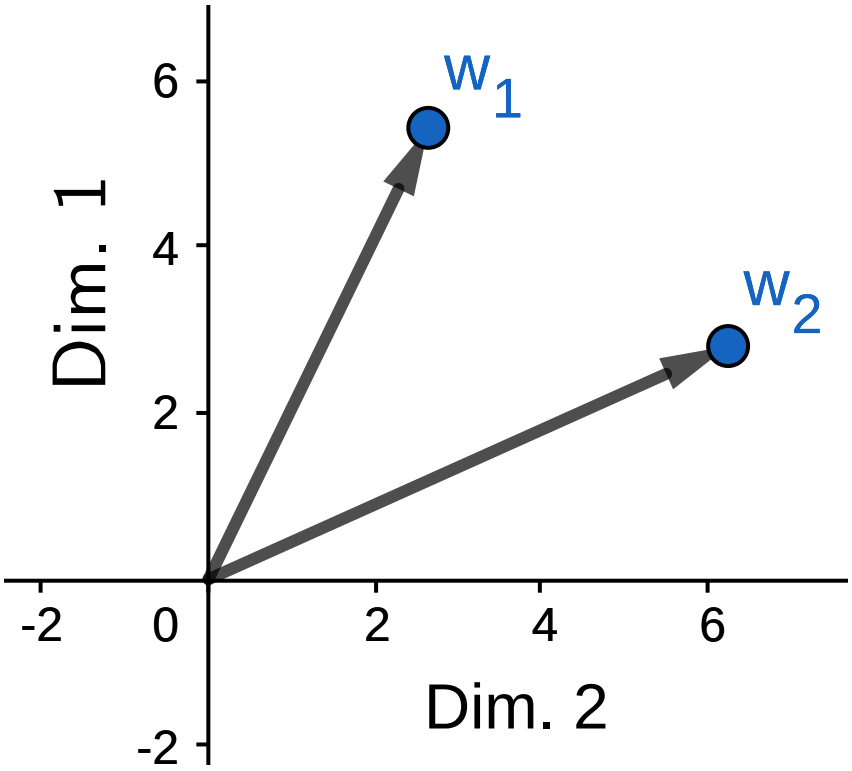


Figura 2.6: Representación del embebido de palabras bajo $\mathbf{P}$ *de dimensión dos. Gráfico de elaboración propia.*

2.2.3 Modelo GloVe

Pennington et al. (2014) propuso un trabajo en la misma dirección que Mikolov: vectorizar las palabras. Sin embargo, su enfoque parte de una matriz de co-ocurrencias $\mathbf{X}$ donde sus entradas x_{ij} es el número de veces que la j-ésima palabra, w_j del vocabulario aparece en el contexto de la palabra i-ésima, w_i. De esta forma, la probabilidad de que la w_j ocurra en el contexto de w_i puede calcularse como sigue:

$$P_{i,j} = \frac{x_{ij}}{x_{i.}}$$

Más aún, Pennington y su equipo hacen las siguientes observaciones:

- Das tres palabras w_{i_1}, w_{i_2} y w_{i_3}. Si w_{i_3} está más relacionada con w_{i_1} que con w_{i_2}. Entonces $P_{i_1,i_3} > P_{i_2,i_3}$. Por lo cuál, se espera que el cociente $P_{i_1,i_3}/P_{i_2,i_3}$ sea grande.

[11]Hasta el 2021 no existe no existe trabajos que den razón del porqué el enfoque de Mikolov refleja lo mencionado en el trabajo de Harris hace ya cerca de siete décadas.

Por otro lado, si w_{i_3} está más relacionada con w_{i_2} que con w_{i_1} entonces se esperaría que el cociente $P_{i_1,i_3}/P_{i_2,i_3}$ sea pequeño.

- Debe existir alguna función $F\left(w_{i_1}, w_{i_2}, w_{i_3}\right)$ tal que pueda recuperar dichos cocientes.
- Esa misma función debe poder capturar el contexto dado por la palabra w_{i_3} y rescatar la diferencia entre las palabras mismas. Por ello, resulta sensato proponer que

$$F\left(w_{i_1}, w_{i_2}, w_{i_3}\right) = F\left(\left(w_{i_1} - w_{i_2}\right)' w_{i_3}\right) = \frac{F\left(w'_{i_1} w_{i_3}\right)}{F\left(w'_{i_2} w_{i_3}\right)}$$

Finalmente, el modelo GloVe (Global Vectors) es el par ordenado de matrices $\left(\mathbf{W}, \tilde{\mathbf{W}}\right)$ tal que:

$$\ln x_{ij} = \mathbf{w}'_i \tilde{\mathbf{w}}_k + b_i + \tilde{b}_k \tag{2.4}$$

donde $\mathbf{w}_i$ es la i-ésima columna de $\mathbf{W}$, $\tilde{\mathbf{w}}_k$ la k-ésima columna de $\tilde{\mathbf{W}}$, b_i es una constante de sesgo asociada a la representación de $\mathbf{w}_i$ y, $\tilde{b}_k$ es una constante de sesgo asociada a la representación del contexto de $\tilde{\mathbf{w}}_k$. Expresión que puede resolverse mediante mínimos cuadrados mediante la siguiente función de error:

$$E = \sum_{i,j} f\left(x_{ij}\right) \left(\mathbf{w}'_i \tilde{\mathbf{w}}_k + b_i + \tilde{b}_k - \ln x_{ij}\right)^2 \tag{2.5}$$

donde f es una función de penalización dada por:

$$f(x) = \begin{cases} \left(\frac{x}{100}\right)^{3/4} & \text{si } x < 100 \\ 1 & \text{o.c} \end{cases} \tag{2.6}$$

Shi y Liu (2014) mencionan que, en la practica, es altamente probable que se tengan varias entradas $x_{ij} = 0$ por lo que resulta conveniente sustituir x_{ij} por $x_{ij} + 1$. Además de que, las constantes de sesgo no se emplean en la optimización dejando su obtención a un paso posterior empleando algún algoritmo de descomposición de matrices.

2.2.4 Sub-palabras

2.3 Pre-procesamiento

Chomsky (1974) menciona que es imposible, solo recolectando locuciones, descubrir todos los patrones existentes en el lenguaje. En ese sentido, lo único que queda es apostar por el volumen. Sin embargo, la cantidad de locuciones revisadas es relativamente pequeña en comparación con la cantidad de locuciones no revisadas que abundan en la red. Por esta razón muchos trabajos han mirado mirado a la web y en particular a las redes sociales, dentro de las cuales resalta Twitter. Sin embargo, de acuerdo con Clark y Araki (2011), la irregularidad de las locuciones hace difícil su uso en aplicaciones de PLN y es por ello que resulta necesario pre-procesar la información.

Las tareas de pre-procesamiento varían en función de las irregularidades de la fuente. En este trabajo, se empleó con la información de la red social Twitter. En ese sentido, Billal et al. (2016) mencionan que el pre-procesamiento consiste en tres tareas

fundamentales: Normalización, Segmentación y Reconocimiento de Entidades Nombradas (REN).

2.3.1 Normalización

La normalización consiste en darle regularidad a las palabras. En redes sociales es común que para hacer énfasis se repitan algunos de los caracteres que conforman la palabra. Por ejemplo, en lugar de escribir *mucho* se escriba *muuuuucho*. Otro problema con la información obtenida de las redes sociales es la eliminación de las vocales o el cambio de algunos caracteres. Por ejemplo, en lugar de escribir *porque* solo se escribe *pq*. Otro problema clásico son los errores de ortográficos. Por ejemplo, escribir *decisión* en lugar de *decisión*. Sin embargo, de acuerdo con los resultados obtenidos por Hassan y Menezes (2013), el problema principal consiste en detectar si en efecto se está ante la presencia de una irregularidad o no. Así, la normalización consiste en una especie de suavizamiento del texto.

Problemas por énfasis y expresiones regulares

Uno puede pensar en la Gramática Generativa de Chomsky en una especie de meta-lenguaje, en el sentido de que, como ya se había mencionado anteriormente, los elementos del lenguaje que propone son oraciones y el alfabeto son los morfemas. Pero note que, a su vez los morfemas con pueden ser elementos de otro lenguaje (una especie de sub-lenguaje) donde, su alfabeto son los caracteres como las letras del abecedario latino.

En la práctica computacional, se sustituye a los morfemas por palabras completas. Así, el lenguaje es una colección de palabras, en lugar de morfemas como proponía Chomsky. De esta forma, en un sentido estricto, el hacer énfasis repitiendo caracteres lleva a que, por ejemplo, palabras como *mucho*, *muuucho* y *muuuuuchooooooo* formen parte del lenguaje. Sin embargo, humanamente todas estas palabras son mapeadas a una única palabra *mucho*. Por lo tanto, el objetivo es poder describir la regularidad presente en el énfasis para luego, hacer el mapeo correspondiente a esa única palabra.

Un de los usos de las expresiones regulares es la realización de tareas como la mencionada en el párrafo anterior. Goyvaerts y Levithan (2012) definen una expresión regular como un patrón especifico de texto, es otras palabras, una regularidad en el lenguaje. Formalmente, en teoría de la computación, de acuerdo con Cohen (1991) una expresión regular es un subconjunto A de palabras de un lenguaje con un alfabeto Σ que puede ser expresadas mediante una serie finita de las siguientes operaciones:

1. *Concatenación.* Dadas dos cadena de caracteres s_1 y s_2 formadas por una cantidad finita de caracteres del alfabeto Σ. Se define la concatenación de éstas como

$$s_1 s_2 = \{s_1 s_2\}$$

2. *Unión.* Dadas dos cadena de caracteres s_1 y s_2 formadas por una cantidad finita de caracteres del alfabeto Σ. Se define la unión de éstas como

$$s_1 + s_2 = \{s_1, s_2\}$$

3. *Estrella de Kleene.* Dada una cadena de caracteres s formada por una cantidad finita de caracteres del alfabeto Σ. Se define la estrella de Kleene de s como el

siguiente conjunto

$$s* = \{\Lambda, s, ss, sss, \ldots\}$$

4. *Estrella positiva de Kleene.* Dada una cadena de caracteres s formada por una cantidad finita de caracteres del alfabeto Σ. Se define la estrella de Kleene de s como el siguiente conjunto

$$s^+ = \{s, ss, sss, \ldots\}$$

De esta manera, es posible describir la regularidad de énfasis de la palabra *mucho* como sigue:

$$mu^+cho^+ = \{mucho, muchoo, \ldots, muucho, muuchoo, \ldots, muuucho, muuuchoo, \ldots\}$$

Sin embargo, la implementación de expresiones regulares dentro de un lenguaje de programación tiene algunas particularidades extras. En Python, el uso de expresiones regulares se maneja en dos niveles, a nivel de carácter y a nivel de expresión regular.

Nivel de carácter. En Python, se dice que se trabaja a nivel de carácter cuando se definen lo que se conocen como clases de carácter. Para definir una clase de carácter se emplean corchetes. Así, dados dos caracteres del alfabeto, c_1 y c_2, se tiene las siguientes equivalencias:

$$[c_1c_2] = \{c_1, c_2\}$$

$$[\hat{}\, c_1c_2] = \{c_1, c_2\}^c$$

Note que las clases de caracteres son subconjuntos de Σ. Para facilitar la descripción de clases de carácter, Python permite definir de manera corta la clase de minúsculas, mayúsculas y dígitos como:

$$[abcd \ldots z] = [a - z]$$

$$[ABCD \ldots Z] = [A - Z]$$

$$[0, 1, 2 \ldots 9] = [0 - 9]$$

Además, de que existen siete clases de carácter predefinidas y que se muestra en la Tabla 2.1.

Tabla 2.1: Expresiones predefinidas en Python. Tabla de elaboración propia con base en lo expuesto en la obra de López y Romero (2014).

Forma corta	**Forma larga**
.	$[\hat{}\, \backslash n]$
\d	$[0 - 9]$
\D	$[\hat{}\, 0 - 9]$
\s	$[\backslash t \backslash n \backslash r \backslash f \backslash v]$
\S	$[\hat{}\, \backslash t \backslash b \backslash r \backslash f \backslash v]$
\w	$[a - zA - Z0 - 9]$
\W	$[\hat{}\, a - zA - Z0 - 9]$

Adicional a esto, Python reconoce 4 operaciones generales que puede aplicarse tanto a un carácter como a una clase de carácter:

- **Estrella de Kleene.** La estrella de Kleene se sigue representando por el símbolo asterisco (*).
- **Estrella positiva de Kleene.** La estrella positiva de Kleene se sigue representando por el símbolo de suma (+).
- **Operación cerradura de pregunta.** En Python símbolo de cerradura de pregunta (?) indica que ese carácter o clase de carácter puede aparecer o no.
- **Rango de repeticiones.** En Python puede agregarse al final de un carácter o una clase de carácter {} para indicar ciertas repeticiones del carácter o la clase. En ese sentido, $\{n\}$ indica una cantidad fija de repeticiones, $\{n,\}$ indica una cantidad mínima de repeticiones, $\{,n\}$ una cantidad máxima de repeticiones y, $\{n,m\}$ un rango de repeticiones.

Nivel de expresión regular. Para definir una expresión regular en Python esta se escribe entre diagonales como sigue:

$$r'\text{Expresión regular}'$$

Además para dado que el texto de búsqueda puede ser extenso, Python agrega 6 delimitadores de búsqueda, los cuales se muestran en la Tabla 2.2.

Tabla 2.2: Delimitadores de expresiones regulares. Tabla de elaboración propia con base en lo lo expuesto en la obra de López y Romero (2014).

Notación en Python	Significado
^	Inicio de línea
$	Fin de línea
\b	Frontera de palabra
\B	Todo lo que no es frontera de palabra
\A	Inicio de la entrada
\Z	Fin de entrada

Para dar claridad sobre el uso de los delimitadores en las expresiones regulares, considere la siguiente entrada:

Hola Juanito Hola
2Hola Hola Pedrito Hola

Así se tiene lo siguiente:

- La expresión regular *r'Hola'* hará correspondencia con los todos los *Hola*.
- La expresión regular *r'^(Hola)'* solo hará correspondencia con el primer *Hola*.
- La expresión regular *r'(Hola)$'* solo hará correspondencia con el segundo y último *Hola*.
- La expresión regular *r'\bHola'* hará correspondencia con todos los *Hola*, excepto el tercero.

- La expresión regular *r'\BHola'* solo hará correspondencia con el tercer *Hola*.
- La expresión regular *r'\A(Hola)'* solo hará correspondencia con el primer *Hola*.
- La expresión regular *r'(Hola)\Z'* solo hará correspondencia con el último *Hola*.

Generalidades. Sea cual sea el nivel de descripción, en Python, debido a la notación existen 12 símbolos que no significan el símbolo de forma literal: Diagonal inversa (\), intercalación (^), símbolo de dólar ($), punto (.), barra vertical (|), cierre de pregunta (?), asterisco (*), símbolo de suma (+), apertura de paréntesis ('('), cerradura de paréntesis (')'), apertura de corchete ([) y apertura de llave ({)[12].

Corrección automática de errores

Hládek et al. (2020) mencionan que un algoritmo de corrección de errores consta de tres elementos: diccionario, modelo del error y modelo de contexto, siendo éste último opcional. La idea general es emplear el diccionario para generar candidatos y acomodarlos de acuerdo con el modelo de error y el modelo de contexto.

Modelo del error. Consiste en evaluar cuál es la viabilidad de cambiar una palabra incorrecta por alguna palabra correcta del diccionario. La viabilidad en este caso se cuantifica por medio de la distancia entre la palabra incorrecta y las palabras del diccionario. Así, dos palabras son igualmente viables como posibles correcciones si su distancia a la palabra incorrecta es la misma. Para ello, Hládek et al. (2020) mencionan que se emplean principalmente las siguientes funciones de distancia: LD, DLD, LCS las cuales se conocen como métricas o distancias de edición.

Damerau (1964) menciona una forma de determinar es viable corregir una cadena, s_1, por otra cadena, s_2, es mediante el empleo de la distancia de Hamming. En este sentido, la corrección es viable si sus longitudes son iguales y la distancia de Hamming entre ambas es a lo sumo 1. Waggener et al. (1995) definen la distancia de Hamming para cadenas como sigue:

Definición 3 (Distancia de Hamming) *Sean* $s_1 = c_{s_1}^1 c_{s_1}^2 \dots c_{s_1}^n$ *y* $s_1 = c_{s_2}^1 c_{s_2}^2 \dots c_{s_2}^n$ *dos cadenas formadas por n caracteres. Entonces, la distancia de Hamming entre las cadenas está dada por:*

$$d(s_1, s_2) = \sum_{i=1}^{n} I\left(c_{s_1}^i \neq c_{s_2}^i\right) \tag{2.7}$$

Por ejemplo, *menestro*[13] pudiera ser sustituido por *ministro*, *menester* o *maestros*. Luego, empleado (2.7)

$$d(\text{menestro}, \text{ministro}) = 2$$

$$d(\text{menestro}, \text{menester}) = 2$$

$$d(\text{menestro}, \text{maestros}) = 7$$

[12]Como nota curiosa en LaTeX la diagonal inversa, la intercalación y el símbolo de dólar también son símbolos prohíbidos

[13]La siguiente publicación en Twitter `https://twitter.com/Justaceo1/status/1379135241898590210` posee el uso *menestro*.

Por lo que los tres candidatos pueden ordenarse como sigue,

$$\text{ministro} = \text{menester} > \text{maestros}$$

Por otro lado, Levenshtein (1966) identifica tres fuentes de ruido en las cadenas: inserción, eliminación e inversión. La inserción consiste de agregar algún carácter en alguna posición dentro de la cadena original; la eliminación, la operación inversa de la inserción, es la eliminación de algún carácter de la cadena y; la inversión que consiste en permutar una subcadena de la cadena. En ese sentido se define:

Definición 4 (Distancia LD) *Sean s_1, s_2 dos cadenas la distancia LD entre ambas cadenas es el número de inserciones, eliminaciones e inversiones mínimas necesarias para transformar s_1 en s_2.*

De esta forma, bajo la distancia LD se tiene que:

$$d(\text{menestro}, \text{ministro}) = 4$$

$$d(\text{menestro}, \text{menester}) = 3$$

$$d(\text{menestro}, \text{maestro}) = 3$$

$$d(\text{menestro}, \text{maestros}) = 4$$

Por lo que los cuatros candidatos pueden ordenarse como sigue,

$$\text{menester} = \text{maestro} > \text{ministro} = \text{maestros}$$

Note que *maestro* ahora puede ser candidato porque en este caso no existe una condición sobre la longitud.

Una tercera alternativa consiste en reconocer una operación adicional a las tres propuesta por Levenshtein: la sustitución. Debido a que, técnicamente, ésta adición es equivalente a mezclar la propuesta de Damerau y Levenshtein, a esta alternativa se le conoce como distancia DLD (Damerau-Levenshtein Distance). Bajo esta alternativa

$$d(\text{menestro}, \text{ministro}) = 2$$

$$d(\text{menestro}, \text{menester}) = 2$$

$$d(\text{menestro}, \text{maestro}) = 2$$

$$d(\text{menestro}, \text{maestros}) = 3$$

Por lo que los cuatros candidatos pueden ordenarse como sigue,

$$\text{menester} = \text{maestro} = \text{ministro} > \text{maestros}$$

Finalmente, otro criterio para ordenar los candidatos es el criterio de subcadena común más larga con k errores bajo la distancia de Hamming descrito por Flouri et al. (2015).

Definición 5 (Subcadena más larga común) *Sean s_1 y s_2 cadenas de longitud n y m con $n \leq m$. Dado un entero k y $\phi(s, i, j)$ subcadena que va del i-ésimo carácter al j-ésimo carácter de una cadena s con $i < j$. La subcadena común más larga con k de s_1 y s_2, ϕ^* es tal que:*

$$\phi^* = \max_{i,j} \phi(s_1, i, j) \quad s.a \quad d(\phi(s_1, i, j), \phi(s_2, i, j)) \leq k$$

donde la distancia es la distancia de Hamming.

Por ejemplo, si $k = 2$ se tiene que la subcadena más larga con 2 errores de *menestro* respecto *ministro* es *menestro*, respecto *menester* es *menestro*, respecto *maestro* es *men* y, respecto a *maestros* es *men*. Por ende los candidatos pueden ordenarse como:

$$\text{menester} = \text{ministro} > \text{maestro} = \text{maestros}$$

Modelo de de contexto. En el ejemplo de la palabra *menestro*, el contexto:

> *mucha vehemencia con las faltas de ortografía de la consejera de Vox en Murcia pero el otro día con la cagada del* ***menestro*** *de Universidades...*

deja entrever que la palabra más adecuada es *ministro*. Los modelos de contexto, se encargan de asignar una segunda calificación a los candidatos con base en el contexto, el cual, usualmente está compuesto por las palabras que anteceden al término a corregir.

Hládek et al. (2020) menciona que los modelos de contexto empleados para corrección de errores son las GCL, modelos basado en n-gramas, Modelos Ocultos de Markov (MOM), técnicas de aprendizaje máquina como Árboles de Desición, Máquinas de Vectores de Soporte (MVS) o Redes Neuronales (RN). Sin embargo, la herramienta usada varía de lenguaje a lenguaje. En el caso del español, Hládek et al. (2020) solo identifican cuatro trabajos, de los cuales solo uno de ellos está enfocado al pre-procesamiento de texto proveniente de redes sociales, el trabajo de Melero et al. (2016).

Melero y su equipo proponen, para el caso del español, un modelo mixto que consiste en la ponderación de cuatro modelos basados en 3-gramas, el cuál está dado por la siguiente expresión:

$$f(s, w^*) = \lambda_1 f_{TC}(s, w^*) + \lambda_2 f_{LC}(s, w^*) + \lambda_3 f_{Lem}(s, w^*) + \lambda_4 f_{PoS}(s, w^*) \tag{2.8}$$

donde s es la cadena a corregir[14] y w^* es una palabra candidato, $\lambda_i > 0$, $\sum_{i=1}^{4} \lambda_i = 1$ y

$$f_T(s, w^*) = \sum_i P\left[w^* \in C^i_{W_n} | W_{n-1} \in C_{W_{n-1}}, W_{n-2} \in C_{W_{n-2}}\right] P\left[W_{n-1} \in C_{W_{n-1}} | W_{n-2} \in C_{W_{n-2}}\right] P\left[W_{n-2} \in C_{W_{n-2}}\right] \tag{2.9}$$

donde $T = TC, LC, Lem, PoS$ es el nivel del modelo, C_W es la clase de la palabra relativa a t. Donde el nivel TC se refiere al texto sin transformar, LC al texto en minúsculas, Lem a la forma lematizada del texto y PoS a la forma etiquetada del texto. Por ejemplo, el caso de la cadena *menestro* bajo el modelo de 3-gramas las palabras contexto son *cagada* y *del*, así si para la palabra candidata *ministro*, se tiene:

$$f_{TC}(s, w^*) = f_{TC}(s, w^*) = \ P\left[\text{ministro}|\text{del}, \text{cagada}\right] P\left[\text{del}|\text{cagada}\right] P\left[\text{cagada}\right]$$

En los primeros dos niveles las clases son las mismas palabras, solo que a nivel TC en uno se mantiene la capitalización del texto original y, a nivel LC se pasa todo el texto a minúsculas. Por otro lado, para el nivel Lem se tiene:

$$f_{Lem}(s, w^*) = \ P\left[\text{ministro}|\text{del}, \text{cagar}\right] P\left[\text{del}|\text{cagar}\right] P\left[\text{cagar}\right]$$

[14]Se hace la distinción entre palabra y cadena, en el entendido que toda palabra es una cadena pero, no toda cadena es una palabra.

Mientras que a nivel PoS se tiene:

$$f_{PoS}(s, w^*) = P\left[\text{sustantivo}|\text{determinante}, \text{sustantivo}\right] P\left[\text{determinante}|\text{sustantivo}\right] P\left[\text{sustantivo}\right]$$

Sin embargo, Melero y su equipo estiman que $\lambda_4 = 0$, $\lambda_3 = 0.0962$ y $\lambda_1 = \lambda_2 = 0.4519$. Con lo que (2.8) puede reducirse a:

$$f(s, w^*) \approx 0.4519 f_{TC}(s, w^*) + 0.4519 f_{LC}(s, w^*) \tag{2.10}$$

2.3.2 Segmentación

En Twitter parte de la información está contenida en lo que se denomina *hashtag*, los cuales organizan la información por temas o eventos (Small, 2011). Por ejemplo, el tweet:

> #Anticonvocatoria donde nada #CambiaElJuego todo lo que escucho es #AMLOLujoDePresidente efímero y devaluado igual #Letrinus y todos los demás.
>
> -*@causa_nuestra*[15]

contiene cuatro *hashtags* de los cuáles dos tiene información que debe ser segmentada: *Cambia el juego* y *AMLO lujo de presidente*. En ese sentido, el algoritmo más usado de segmentación es el algoritmo basado en uni-gramas y 2-gramas propuesto por Norvig (2009). En su trabajo Norvig propone dada una cadena s_0 se puede segmentar como $s_0 = w_1 s_1$, donde a su vez se puede segmentar $s_1 = w_2 s_2$ y, así sucesivamente, donde s_{i+1} es la cadena sobrante al quitar w_{i+1} de la cadena s_i y

$$w_{i+1} = \arg\max_{w^*} P\left[w_{i+1}|w_i\right] P\left[w_i\right] P[s_{i+1}] \tag{2.11}$$

donde $P[s_{i+1}]$ funciona como un ponderador y las probabilidades se calculan usando la distribución de frecuencias del *corpus*. Sin embargo, como en la mayoría de los casos las cadenas s_i no son palabras, entonces Norvig propone un modelo probabilístico para palabras desconocidas dada por la siguiente expresión:

$$P[s] = \frac{10}{|s|10^{|s|}} \tag{2.12}$$

Sin embargo, Norvig comenta que la expresión anterior no tiene ningún sustento teórico y se debe a un ajuste empírico. En ese sentido, Cano-Felix (2019) encontró que, para el caso del español, una mejor alternativa es el uso de un modelo de probabilidad Weibuill discreta de parámetros $\alpha = 7.5291$ y $\beta = 2.2066$ donde la variable aleatoria sigue estando en términos de la longitud de la cadena.

2.3.3 Reconocimiento de Entidades Nombradas

El REN consiste en identificar sustantivos propios, algunas abreviaturas como el tipo de moneda o fechas en el texto. Los nombre de los países, ciudades y nombres propios

[15]El tweet puede ser consultado en `https://twitter.com/causa_nuestra/status/1377178856193359874`

son ejemplos de entidades nombradas. Usualmente, éstas están compuestas por una sola *palabra*, como *Mexicali*, *México* o *Juan*. Sin embargo, no todas las entidades nombradas son de una palabra. Por ejemplo:

> Cierto. Algo tuve que ver representando a un **Instituto Universitario Tecnológico**.
>
> -*@castellimika*[16]

En este sentido *Instituto Universitario Tecnológico* es una entidad nombrada. Dentro del texto formal, las entidades nombradas tienen la característica de estar capitalizadas; pero dentro del texto informal esto no siempre ocurre. Por ejemplo, en el siguiente tweet además de otros errores se puede apreciar que la entidad nombrada *Mexicali* no está capitalizada.

> Otro gran lunes como cada semana #LaMesaRenonaEnVivo saludos @franco_esca @elchristianmeza @lamolechida desde **mexicali** por otra gran noche de risas #soytoronto y tambien saludos para el rubiiiiiiiiiii me encanta que le cagues la vida al @elchristianmeza
>
> -*@bambino03garcia* [17]

Nouvel et al. (2016) reconocen que existen cinco enfoques para el REN, cuatro supervizados: Naïve Bayes, MOM, Redes Neuronales y los Campos Aleatorios Condicionales (CAC) y uno no supervizado basado en semántica. Siendo los CAC de Cadena Lineal el enfoque con mejores resultados en la tarea.

Debido a la relación que tienen los CAC con los MOM, se empezarán a describir los MOM primero. Ross (2014) define estos como sigue:

Definición 6 (Modelo de Markov Oculto) *Sea* $\{X_i, i = 1, 2, \ldots\}$ *una cadena de Markov con probabilidades de transición* P_{ij} *y probabilidades de estado inicial* p_i*. Suponga también que, existe un conjunto finito de señales* $\mathcal{S}$ *que se emite cada que la cadena de Markov entra a un estado. Además, considere que cuando la cadena de Markov entra a un estado* j *entonces, independientemente del estado previo de la cadena y la señales previas, la probabilidad de que la señal emitada sea* s *solo depende del estado actual* j*, es decir,* $p(s|j)$ *de esta forma si* S_i *denota la i-ésima señal emitida se tiene que:*

$$p(s|j) = P\left[S_n | X_1, S_1, X_2, S_2 \ldots, X_{n-1}, S_{n-1}, X_n = j\right]$$

En ese sentido, note que las señales $S_1, S_2 \ldots$ *serán una secuencia de estados observable mientras la cadena* $X_1, X_2 \ldots$ *no lo será, siendo está una cadena de Markov oculta, y al modelo de probabilidades de transición se le conoce como Modelo de Markov Oculto (MOM).*

---

[16]El tweet puede ser consultado en `https://twitter.com/castellimika/status/1379226926028587009`

[17]El tweet puede ser consultado en `https://twitter.com/bambino03garcia/status/1379275928233107458`

Desde el punto de vista del NER, las señales son la palabras, W y, los estados de la cadena no observable las etiquetas asociadas al NER, S (Fig. 2.7). Note que, adecuando la definición al contexto del NER, se tiene que:

$$p(s|j) = P\left[W_{L+1} = w_j | S_1, W_1, S_2, W_2 \ldots, S_L, W_L, S_L = s_j\right]$$

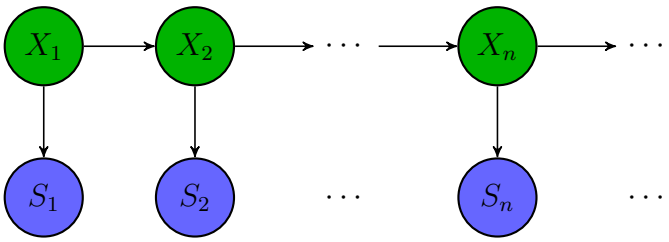
Aunque, desde el punto de vista de la aplicación se está interesado en calcular para $k \leq L$ la probabilidad de que la k-ésima entidad sea de la clase s_j usando la información dada por las señales, es decir:

$$P\left[S_k = s_j | S_1 = s_{i_1}, S_2 = s_{i_2}, \ldots, S_L = s_{i_L}\right] \tag{2.13}$$

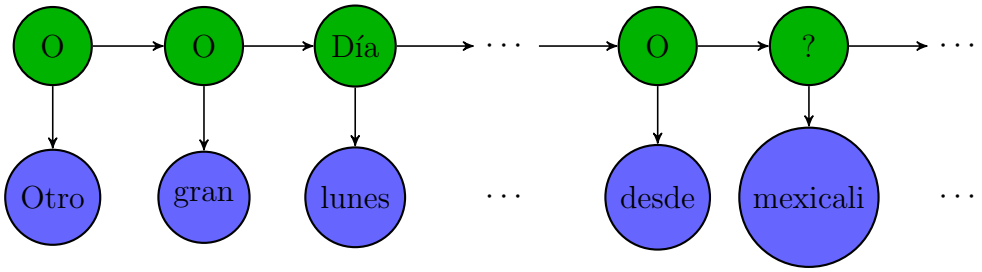
Nouvel et al. (2016) menciona que, a menudo en el contexto de la NER, se acostumbra calcular de probabilidad de que una sucesión de entidades tenga ciertas etiquetas, en lugar de lo planteado en (2.13. De esta forma, si se conocen las etiquetas de las primeras k entidades, la idea es encontrar las clases o etiquetas $s_{j_1}, s_{j_2}, \ldots, s_{j_L}$ tales que se maximice

$$P\left[S_1 = s_{j_1}, \ldots, S_L = s_{j_L} | W_1 = w_{i_1}, \ldots, W_L = w_{i_L}\right] \tag{2.14}$$

dado que, como ya se mencionó se conocen los primeros k estados de la cadena no observable.



(a) Diagrama de un Modelo de Markov Oculto. Gráfico de elaboración propia.



(b) Diagrama de un Modelo de Markov Oculto en el contexto del NER aplicado al segundo tweet de ejemplo en este apartado. En este caso, la etiqueta 'O' significa 'Otro' para denotar que no es una entidad nombrada. Gráfico de elaboración propia.

Figura 2.7: Diagrama de Modelo Oculto de Markov, en verde se muestra la cadena de Markov oculta y en azul las señales observables.

La expresión planteada por (2.14) usualmente se resuelve recursivamente empleando el algoritmo de Viterbi. En lo que respecta a los CAC de Cadena Lineal, estos son una generalización de los MOM. Para ello, note que la expresión (2.14) por definición de probabilidad condicionada es equivalente a:

$$P\left[S_1 = s_{j_1}, \ldots, S_L = s_{j_L} | W_1 = w_{i_1}, \ldots, W_L = w_{i_L}\right] = \frac{P\left[S_1 = s_{j_1}, \ldots, S_L = s_{j_L}, W_1 = w_{i_1}, \ldots, W_L = w_{i_L}\right]}{P\left[W_1 = w_{i_1}, \ldots, W_L = w_{i_L}\right]} \tag{2.15}$$

La cual por la propiedad markoviana, la independencia de las señales puede reescribirse como y aplicando marginalización:

$$P\left[S_1 = s_{j_1}, \ldots, S_L = s_{j_L} | W_1 = w_{i_1}, \ldots, W_L = w_{i_L}\right] = \frac{\prod_{h=1}^{n} P\left[S_h = s_{j_h} | S_{h-1} = s_{j_{h-1}}\right] P\left[W_h = w_{i_h} | S_h = s_{j_h}\right]}{\sum_{\mathbf{s}} \prod_{h=1}^{n} P\left[S_h = s_{j_h} | S_{h-1} = s_{j_{h-1}}\right] P\left[W_h = w_{i_h} | S_h = s_{j_h}\right]} \tag{2.16}$$

Definiendo $\theta_{j_i,j_{i-1}} = \ln P\left[X_i = j_i | X_{i-1} = j_{i-1}\right]$ y $\mu_{s_i,j_i} = \ln P\left[S_i = s_i | X_i = j_i\right]$ la expresión anterior puede volver a escribir como:

$$P\left[X_1 = j_1, \ldots, X_n = j_n | S_1 = s_1, \ldots, S_n = s_n\right] = \frac{\prod_{i=1}^{n} \exp\left\{\theta_{j_i,j_{i-1}} + \mu_{s_i,j_i}\right\}}{\sum_{\mathbf{s}} \prod_{i=1}^{n} \exp\left\{\theta_{j_i,j_{i-1}} + \mu_{s_i,j_i}\right\}} \tag{2.17}$$

Finalmente, esta última expresión puede escribirse como:

$$\frac{\prod_{i=1}^{n} \exp\left\{\sum_{j_i,j_{i-1} \in J} \theta_{j_i,j_{i-1}} I\left(X_i = j_i\right) I\left(X_{i-1} = j_{i-1}\right) + \sum_{j_i \in J} \sum_{s_i \in S} \mu_{s_i,j_i} I\left(X_i = j_i\right) I\left(S_i = s_i\right)\right\}}{\sum_{\mathbf{s}} \prod_{i=1}^{n} \exp\left\{\sum_{j_i,j_{i-1} \in J} \theta_{j_i,j_{i-1}} I\left(X_i = j_i\right) I\left(X_{i-1} = j_{i-1}\right) + \sum_{j_i \in J} \sum_{s_i \in S} \mu_{s_i,j_i} I\left(X_i = j_i\right) I\left(S_i = s_i\right)\right\}} \tag{2.18}$$

Por otro lado, Sutton y McCallum (2019) define un CAC de Cadena Lineal como sigue:

Definición 7 (Campo Aleatorio Condicional de Cadena Lineal) *Sean* $\mathbf{S}$ *y* $\mathbf{X}$ *vectores aleatorios,* $\theta \in \mathbb{R}^K$ *un vector de parámetros y,* $\mathcal{F} = \{f_k(x, x', \mathbf{s}_i)\}_{k=1}^{K}$ *un conjunto de funciones reales, que se llamarán funciones de características. Entonces un Campo Aleatorio Condicional de Cadena Lineal es la condicionada de* $\mathbf{X}|\mathbf{S}$ *tal que esta puede escribirse como:*

$$P[\mathbf{X} = \mathbf{x} | \mathbf{S} = \mathbf{s}] = \frac{1}{Z(\mathbf{x})} \prod_{t=1}^{T} \exp\left\{\sum_{k=1}^{K} \theta_k f_k\left(x_t, x_{t-1}, \mathbf{s}_t\right)\right\} \tag{2.19}$$

con

$$Z\left(\mathbf{x}\right) = \sum_{\mathbf{s}} \prod_{t=1}^{T} \exp\left\{\sum_{k=1}^{K} \theta_k f_k\left(x_t, x_{t-1}, \mathbf{s}_t\right)\right\} \tag{2.20}$$

Note que usando la Def. 7 con K es el total de índices que recorren las sumas planteadas por (2.18), las funciones indicadoras de esa misma función como las funciones de características y $\theta_{j_i,j_{i-1}}$ y μ_{s_i,j_i} como parámetros entonces se tiene que el MOM cumple con la definición de un CAC de Cadena Lineal. Pero, note que la definición permite el uso de funciones de características más elaboradas.

Falta diagrama

2.4 Optimización

La estimación de los modelos y tareas de pre-procesamiento mencionadas en apartados anteriores implican una tarea de optimización. En este apartado describimos los métodos de optimización que fueron empleados en este trabajo, aunque la solución del problema de optimización no está de ninguna manera limitado al uso único de las tareas mencionadas.

2.4.1 Estimación de los modelos CBOW y Skip-Gram

Mikolov et al. (2013a) describe que la tarea de estimación usando directamente el algoritmo propagación hacia atrás puede resultar muy costoso computacionalmente. Para ello, propone reducir la carga operacional mediante la aplicación de uno de los siguientes dos enfoques: muestreo negativo y optimización jerárquica.

Muestreo negativo. De acuerdo con Rong (2014) la función de pérdida del modelo CBOW está dada por la siguiente expresión:

$$E = -\mathbf{W}'_{L+1}\mathbf{P} + \ln\left(\sum_{i=1}^{V}\exp\left\{\mathbf{W}'_i\mathbf{P}\right\}\right) \tag{2.21}$$

De acuerdo con Goldberg y Levy (2014), el muestreo negativo consiste en cambiar la función de pérdida planteada por (2.21) por la siguiente expresión:

$$E = -\ln\sigma\left(\mathbf{W}'_{L+1}\mathbf{P}\right) - \sum_{i\in I_{neg}}\ln\sigma\left(\mathbf{W}'_i\mathbf{P}\right) \tag{2.22}$$

donde I_{neg} es un conjunto de índices tal que $I_{neg} \subset \{1, 2, \ldots, V\}$. De esta forma se reducen las operaciones necesarias para aplicar el algoritmo de propagación hacia atrás. Aunque es necesario aclarar que las funciones de optimización no son equivalente, pero de acuerdo con Goldberg y Levy (2014) las representaciones obtenidas son similares a las que se obtendrían usando (2.21).

Optimización jerárquica. El objetivo del modelo CBOW en poder calcular

$$y_j = \frac{e^{u_j}}{\sum_{k=1}^{V}e^{u_k}} \tag{2.23}$$

es decir, la probabilidad de que la palabra siguiente con base en el contexto dado. Matemáticamente,

$$P\left[W_{L+1} = w_j | W_L = w_{i_L}, \ldots, W_{L-N+1} = w_{i_{L-N+1}}\right] \tag{2.24}$$

En ese sentido, el método de optimización jerárquica tiene sustento en el trabajo publicado por Morin y Bengio (2005) que afirma que si el vocabulario puede dividirse en una colección de subconjuntos disjuntos, tal que existe una función c que a cada palabra le asigna con uno de esos subconjuntos $c\left(w_j\right)$ entonces resulta más eficiente calcular (2.24) por medio de la siguiente igualdad:

$$P\left[W_{L+1} = w_j | C = c(w_j), W_L = w_{i_L}, \ldots, W_{L-N+1} = w_{i_{L-N+1}}\right] P\left[C = c(w_j) | W_L = w_{i_L}, \ldots, W_{L-N+1} = w_{i_{L-N+1}}\right] \tag{2.25}$$

De esta forma, un problema donde la etiqueta tiene p niveles se transforma en p problemas de etiquetas binarias, el cuál es más sencillo de resolver.

Algoritmo de propagación hacia atrás. El objetivo del algoritmo de propagación hacia atrás (BP por sus siglas en inglés) consiste en minimizar la función de pérdida de una red neuronal por medio del gradiente de la función de error (Rojas, 1996). En otras, palabras la idea cuantificar las variaciones del error causas por las variaciones en

las entradas. Sin embargo, cuando la arquitectura de una red neuronal posee, al menos, una capa de oculta, no se puede cuantificar directamente.

Por ejemplo, en el caso de la función de pérdida planteada por (2.22) en una primera etapa, por medio del método del gradiente se tiene que:

$$W_{L+1}^{\prime\text{nuevo}} = W_{L+1}^{\prime\text{viejo}} - \eta \frac{\partial E}{\partial \mathbf{W}_{L+1}^{\prime}\mathbf{P}} \cdot \frac{\partial \mathbf{W}_{L+1}^{\prime}\mathbf{P}}{\partial \mathbf{W}_{L+1}^{\prime}} \tag{2.26}$$

Sin embargo, la expresión solo permite actualizar segunda matriz de pesos, para actualizar la primera y dado que la proyección tiene una función de activación lineal, basta ver como varian las proyecciones en función de las entradas. Así, nuevamente por el método del gradiente

$$W_{L+1}^{\text{nuevo}} = W_{L+1}^{\text{viejo}} - \frac{1}{N}\eta \frac{\partial E}{\partial \mathbf{W}_{L+1}^{\prime}\mathbf{P}} \cdot \frac{\partial \mathbf{W}_{L+1}^{\prime}\mathbf{P}}{\partial \mathbf{P}} \tag{2.27}$$

2.4.2 Algoritmo de Viterbi

2.4.3 Algoritmos meta-heurísticos: Evolución diferencial

2.5 Sobre la similitud semántica

Los modelos CBOW, Skip-Gram y GloVe como se mencionó en la sección 2.1 tienen el beneficio de recuperar la estructura semántica de las palabras. Mediante su aplicación cada palabra es representada como un punto en el espacio y entonces dos palabras están relacionadas si la distancia entre sus representaciones es pequeña.

Bellegarda y Monz (2016) mencionan que, lo más común para determinar la similitud semántica entre dos proyecciones $\mathbf{e}_{w_j}$ y $\mathbf{e}_{w_k}$ es mediante cualquiera de las siguientes dos funciones de distancia:

$$d_1\left(\mathbf{e}_{w_j}, \mathbf{e}_{w_k}\right) = \sqrt{\sum_{h=1}^{p}\left(e_{w_j}^h - e_{w_k}^h\right)^2} \quad \text{Distancia Euclideana}$$

$$d_2\left(\mathbf{e}_{w_j}, \mathbf{e}_{w_k}\right) = \frac{\mathbf{e}_{w_j} \cdot \mathbf{e}_{w_k}}{\left\|\mathbf{e}_{w_j}\right\| \left\|\mathbf{e}_{w_k}\right\|} \quad \text{Ángulo entre vectores}$$

donde $e_{w_j}^h$ es la h-ésima entrada de la proyección.

En muchas áreas, el concepto de distancia está relacionado con el concepto de similitud. En ese sentido, la propuesta de este trabajo como se mencionó en el **sección de hipótesis** es observar el comportamiento del embebido bajo otras funciones de distancia, en particular, bajo la distancia de Hausdoff. Entonces, ¿cuál es la distancia de Haussdoff? Munkres (2000) define la distancia de Hausdoff entre dos conjuntos no vacíos A y B de un espacio métrico como:

$$d_H\left(A, B\right) = \text{máx}\left\{\sup_{a\in A} d(a, B), \sup_{b\in B} d(A, b)\right\} \tag{2.28}$$

donde $d(a, B) = \text{ínf}_{b\in B}\, d(a, b)$. Así, considere que cada una de las proyecciones $\mathbf{e}_{w_j}$ y $\mathbf{e}_{w_k}$ es un conjunto de p características e_w^h. Entonces, (2.28) aplicado al contexto de se puede

ver expresare como:

$$d_H\left(\mathbf{e}_{w_j}, \mathbf{e}_{w_k}\right) = \text{máx}\left\{\sup_{1\leq h\leq p} d(e^h_{w_j}, \mathbf{e}_{w_k}), \sup_{1\leq h'\leq p} d(\mathbf{e}_{w_j}, e^{h'}_{w_k})\right\} \tag{2.29}$$

donde $d(e^h_{w_j}, \mathbf{e}_{w_k}) = \text{ínf}_{i\leq h\leq p}|e^h_{w_j} - e^h_{w_k}|$

2.5.1 Silabeo

2.5.2 Sobre aplicaciones terminales

2.6 Teoría lingüística adicional

CAPÍTULO 3

Metodología

3.1 Obtención de locuciones

En la sección 2.1 se mencionó que la clave para poder aproximar la estructura lingüística era por medio de una colección de locuciones. Mientras que, en la sección 2.3 se mencionó que en muchas ocasiones se recurre a las redes sociales para obtener dicha colección. La búsqueda y obtención de locuciones en este trabajo puede resumirse en dos pasos:

1. Determinación de las locuciones a buscar
2. Búsqueda de las locuciones especificadas.

Determinación de las locuciones a buscar. Al recurrir a las redes sociales, lo primero que debe hacerse es determinar lo que hay que buscar. Para ello, en este trabajo se emplearon los datos proporcionados por Real Academia Española: Banco de datos (CREA) (2008)[1] que, de cuerdo con Real Academia Española (RAE) consta de una amplía variedad de textos de habla hispana desde 1975 hasta 2004 y una lista de más de 700,000 locuciones del español.

Tabla 3.1: Lista de las 10 palabras más frecuentes en el español de acuerdo con el CREA.

Palabra	**Frecuencia**
de	9,999,518
la	6,277,560
que	4,681,839
el	4,569,652
en	4,234,281
y	4,180,279
a	3,260,939
los	2,618,657
se	2,022,514
del	1,857,225

El CREA proporciona la lista de locuciones y la frecuencia de aparición dentro de un conjunto de 140,000 documentos analizados (Tabla 3.1). Sin embargo, en la práctica la inclusión de algunas palabras puede ocasionar efectos no deseados en la estimación de los modelos empleados en el PLN. Para ello, desde el nacimiento del PLN se ha hecho empleo de un conjunto llamado *stop-words* o diccionario negativo el consta de una lista

[1] En marzo de 2021 la RAE publica una versión anotada del CREA. Sin embargo contiene una revisión de documentos menos reciente pues solo analiza documentos publicados entre el año 1975 y el año 2000.

de términos que se excluyen en la estimación del modelo para mejorar los resultados obtenidos en tareas de PLN (Fox, 1989). En el caso del español, el trabajo publicado por genediazjr (2016) ha recibido mucha aceptación, por lo que su diccionario negativo de 990 palabras fue el que se empleó para filtrar algunas de las palabras de la lista original del CREA (Tabla 3.2).

Tabla 3.2: Lista de las 10 palabras más frecuentes en el español de acuerdo con el CREA excluyendo las palabras del diccionario negativo.

Palabra	**Frecuencia**
años	203,027
vida	123,491
gobierno	113,011
país	104,568
mundo	101,745
año	100,790
forma	97,165
caso	96,979
millones	81,999
españa	80,195

Búsqueda de las locuciones especificadas. La búsqueda, al momento de escribir esta tesis, es tan sencilla como buscar en algo en Google. Sin embargo, para poder llegar a estas alturas fue necesario hacer lo siguiente:

1. Configuración del acceso remoto a la red social.

2. Procesamiento de la información obtenida, que consiste en la descarga, lectura de la información y almacenamiento en legible de la información obtenida.

La mayoría de las redes sociales ofrece un acceso remoto a su red social con el fin de apoyar a sus clientes que usualmente son compañías que buscan alguna interacción con los usuarios del la misma. Para ello, Twitter, Inc. (2021) ofrece una opción para desarrolladores abierta a cualquier persona u organización enfocada a cuatro tareas principales: Oir y analizar, lo cual consiste en proporcionar acceso a las publicaciones públicas de los usuarios de la red social; Publicidad, lo cual consiste en proveer herramientas para manejar campañas publicitarias dentro de la red social; Publicación, lo cuál consiste en ofrecer herramientas el manejo de imagen y publicaciones de instituciones dentro de la red social y; Captación, que en un conjunto de herramientas que permite dirigir usuarios hacia ciertos contenidos[2].

Para el acceso remoto de cualquier servicio en línea, los prestadores del servicio permiten a los desarrolladores interactuar con sus servicios por medio de una API (Application Programming Interface), la cuál sirve como intermediario entre el lenguaje que emplea el desarrollador y los sistemas del provedor. En este trabajo se empleó la biblioteca *tweepy* v3.8 creada por hammon38 (2021) que sirve como un intermediario de fácil uso

[2]La cuenta que se empleó en este trabajo fue creada en 2017, época en la cual no hacia falta más que crear una cuenta en la red social y activar la opción de desarrollador. Sin embargo, después del escándalo ocasionado por Cambridge Analytica en 2018, la creación de una cuenta de desarrollador debe tramitarse y esperar autorización.

entre la API de Twitter y Python y posteriormente se empleeo el script *get_expressions.py* disponible en mi página de GitHub personal (ver Prefacio.) para obtener 100 publicaciones para cada una de las locuciones contenidas en la lista del CREA con una frecuencia superior a 30 y que no estuvieran contenidas en el diccionario negativo. En total 100,337 locuciones fueron buscadas y se obtuvieron en la mayoría de los casos 100 publicaciones donde aparecían éstas. Lo que dio un total de de 10,033,700 de publicaciones obtenidas y procesadas, donde cada publicación cuenta con un promedio de 10 palabras, por lo que en total se formó una colección de aproximadamente 100,337,000 elementos[3].

3.2 Entrenamiento de las tareas de pre-procesamiento

Algunas de las tareas de pre-procesamiento necesitan no solo de los ejemplos de uso sino, también requieren de que ese conjunto esté etiquetado. Sin embargo, los conjuntos de datos provenientes de las redes sociales no vienen entrenados. Para ello se empleó lo que se conoce como *Web Scrapping* para automatizar las tareas manuales y debido, a las restricciones impuestas por los servidores de los prestadores de servicio, se recurrió al empleo de la Red Privada Virtual de Kaspersky (VPN, por sus siglas en inglés) para evitar, en medida de lo posible, la interrupción del *Web Scrapping* por una tasa excesiva de peticiones.

Chapagain (2019) define el *Scrapping* como una serie de tareas que consisten en: extraer, copiar, revisar y/o recolectar información y que, cuando la información procede de la web, entonces se agrega el adjetivo de *Web* a *Scrapping.* En el caso de este trabajo se recurrió a extraer información de la RAE y del sitio `http://tulengua.es/silabas` donde está implementado el algoritmo de silabeo propuesto por Hernández-Figueroa et al. (2013) puesto que, una propuesta colateral de este trabajo consistió en la implementación del modelo de la sílaba para ahorrar tiempo en algunas de las tareas.

Los scripts de scrapping pueden ser igualmente encontrados en el proyecto de GitHub y están contenidos en dos archivos por separado:

- `rae_scrapping.py`
- `tulengua_scrapping.py`

Ambos archivos realizan la búsqueda automática usando la biblioteca *selenium*[4] y extraen el código HTML del resultado y posteriormente se emplean expresiones regulares para extraer la información requerida. En el caso de la RAE, después de una inspección manual del código HTML se determinaron las siguientes expresiones regulares para extraer la información:

1. '<'+tag_name+'[^>]*>[^<>]^</'+tag_name+'>'
2. '<'+tag_name+'[^>]*>[^<>]*<sup>'
3. '>[^<]*<'

[3]La información recabada representa aproximadamente tan solo 1.8 GB de información. Sin embargo, si el lector, desea aventurarse a obtener sus propios ejemplos porque 1.8 GB no es poca información en estos día. Debo advertirle, que debido a que Twitter no permite exceder cierto número de peticiones el tiempo estimado para obtener una colección de este tamaño podría llevarle cerca de 4 meses.

[4]Para más información sobre le uso de selenium se puede visitar el sitio `https://selenium-python.readthedocs.io/`

La primera expresión regular permite identificar la etiqueta principal donde se alojan los resultados requeridos que, en este caso son las abreviaturas con las que se determina si una palabra es un sustantivo, verbo, adjetivo, adverbio, etc. La segunda expresión permite identificar la etiqueta principal de algunos casos especiales y la tercera expresión regular se emplean para eliminar el código HTML y dejar solo el resultado deseado en texto plano.

En lo que respecta al segundo sitio, donde se encuentra el algoritmo de silabeo, las expresiones regulares empleados fueron:

1. divide\s en\s[0-9]+\ssílabas:\s[^.]+
2. :.*
3. </?b>
4. /[:\s]+/

La primera expresión permite identificar donde se arroja el resultado; mientras que las expresiones 2-4 se emplean para limpiar la cadena y dejar el resultado en texto plano.

3.3 Algoritmo de silabeo y segmentación

Algoritmo de silabeo

Como se mencionó en la sección 2.5 la mayoría de los algoritmos para el silabeo automático en español son algoritmos basados en reglas. El algoritmo propuesto en este trabajo consiste en que las reglas se puedan reproducir bajo un modelo de una red neuronal.

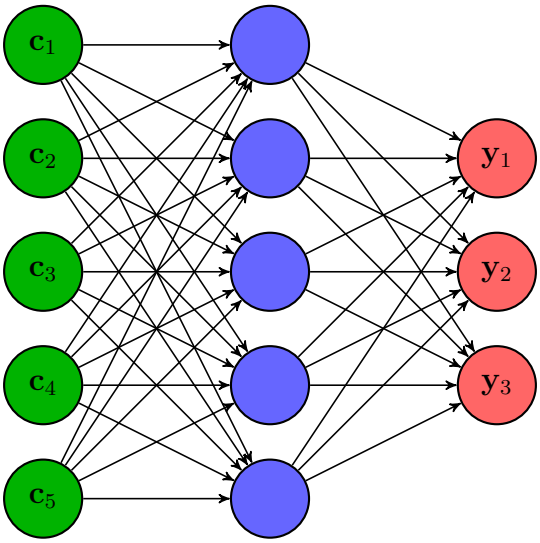


Figura 3.1: Arquitectura propuesta para el modelo de red neuronal que será el núcleo de la silabeo automático.

La Fig. 3.1 propone que el algoritmo emplee el hecho de que una sílaba en español no puede exceder los 5 caracteres. En ese sentido, las entradas del modelo c_i son, son un

conjunto de 5 caracteres donde al menos uno de los caracteres es una vocal. Las salidas y_i obedecen a las probabilidades de que la separación silábica ocurra en la i-ésima posición del bloque. Note que el modelo emula la arquitectura CBOW con la intención que en la capa oculta se condensen las relaciones de los caracteres. Así, en un principio las funciones de activación asociadas a cada capa son las mismas que las empleadas por Mikolov en el modelo CBOW. De esta forma el algoritmo de silabeo propuesto es el siguiente:

Algoritmo 1 Algoritmo de Silabeo

```
1: procedure SYLLABIFICATION(w)                          ▷ w es una palabra
2:     syllables ← []
3:     position ← 0
4:     wc ← CODIFICATION(w)
5:     while postion < len(wc) do
6:         block ← wc[0 : 5]
7:         cut ← CORE_SYLLABIFICATION(block)
8:         syllables ← ADD(syllables, wc[position : (position + cut)])
9:         position ← position + cut
10:    return syllables
```

donde el procedimiento CODIFICATION corresponde a una función de codificación que asigna a cada carácter un valor de acuerdo con la Tabla 3.3 y el procedimiento CORE_SYLLABIFICATION corresponde a la aplicación del modelo de red neuronal descrito en la Fig. 3.1.

Tabla 3.3: Tabla reducida de los sonidos del español con base en la obra de Hualde (2013).

Modo	**Marca**	**Símbolo**	**Grupo**
Sin sonido		h /ʌ/	0
Oclusivo	Bilabial	p b v	1
Oclusivo	Dental	t d	1
Oclusivo	Velar	k q g	2
Fricativo	Labiodental	f	3
Fricativo	Alveolar	s z	4
Fricativo	Palatal	y ll	4
Fricativo	Velar	j g x	5
Africado	Palatal	ch	6
Nasal	Bilabial	m	7
Nasal	Alveolar	n	8
Nasal	Palatal	ñ	8
Aproximante Lateral	Alveolar	l	9
Percusivo	Alveolar	r	10
Vibrante	Alveolar	r rr	10
Vocal fuerte		a e o	11
Vocal débil		i u	12
Semi-vocal		w gü	13

Además, los scripts de procesamiento, entrenamiento, validación e implementación se pueden encontrar en los siguientes archivos del proyecto de GitHub:

- `syllabification_code.py` Script que se encarga de la codificación básica con base en la Tabla .
- `syllabification_train.py` Script que se encarga del entrenamiento de lo que representa el CORE_SYLLABIFICATION. El entrenamiento se realizó usando Keras y Tensorflow.
- `sil_test.py` Script que se encarga de decodificar la información proporcionada por el CORE_SYLLABIFICATION.
- `sil_compare.py` Script que se encarga de la realización de las pruebas de validación, comparando los resultados del método propuesto por Hernández-Figueroa et al. (2013) y el propuesto en este trabajo.
- `sil_model.json` Archivo que contiene la arquitectura del modelo de red neuronal para su rápida carga usando Keras.
- `sil_weights.h5` Archivo que contiene los pesos del modelo de la red neuronal para su uso futuro.
- `sil_xtrain.np` Datos codificados para entrenamiento.
- `sil_ytrain.np` Datos codificados para entrenamiento.
- `syllables_dataset.json` Dados en bruto para entrenamiento.

3.3.1 Algoritmo de segmentación

Cano-Felix (2019) menciona que el proceso de optimización planteado por (2.11) puede ser computacionalmente extensivo cuando se emplean 2-gramas. Sin embargo, también menciona que el método de 2-gramas es mejor que el enfoque más sencillo planteado al usar 1-gramas. En ese sentido, se propone lo siguiente:

- Los cortes en una gran mayoría de los casos aparecen al final de una sílaba[5]. Del conjunto empleado por Cano-Felix (2019) solo el 3 % de los *hashtags* no cumplen esa característica.
- Gai et al. (2014) proponen que en la optimización de (2.11) puede hacerse al generar una segmentación en dos direcciones. Por ejemplo, para separar "elchistedelnorte" la optimización hacia adelante, que es la empleada por Norvig, evalúa a los candidatos *e*, *el*, *elc*, etc. Sin embargo, el proceso puede hacerse hacia atrás donde se evalúan los candidatos *e*, *te*, *rte*,etc.
- Al dividir la cadena en dos subcadenas disjuntas aleatorias es probable que haya al menos una palabra en alguno de las dos subcadenas obtenidas.

El algoritmo de segmentación propuesto y empleado en este consiste en la combinación los tres elementos antes mencionados. Los algoritmos base se describen a continuación y pueden encontrarse condensados en el archivo `segmentation.py`.

[5]Usualmente ocurre así. aunque dar por hecho esto puede llevar a errores sistemáticos. Por ejemplo, uno de los *hashtags* de prueba empleados por Cano-Felix (2019) fue "DíaInternacionaldelamujer" y el silabeo correcto esta dado por *dí-**ain**-ter-na-ci-o-nal-de-la-mu-jer*. Note que la separación correcta ocurre entre la sílaba en negritas, lo que implica su imposible separación.

Algoritmo 2 Algoritmo de Segmentación

procedure SEGMENTATION(s) ▷ s es una cadena
  $blocks \leftarrow$ SYLLABIFICATION(s)
  $k \leftarrow$ RANDOMINT$(0, n)$ ▷ n es el número de sílabas de s
  $b_1, b_2 \leftarrow blocks[0:k], blocks[k:n]$
  $c_{b1A}, c_{b1B} \leftarrow$ SEGMENTATION_FOWARD(b_1), SEGMENTATION_BACKWARD(b_1)
  $c_{b2A}, c_{b2B} \leftarrow$ SEGMENTATION_FOWARD(b_2), SEGMENTATION_BACKWARD(b_2)
  $candidates \leftarrow$ MIX$(c_{b1A}, c_{b1B}, c_{b2A}, c_{b2B})$
  $s_split \leftarrow$ OPTIMIZE$(candidates)$
  return s_split

Algoritmo 3 Algoritmo de Segmentación hacia adelante

procedure SEGMENTATION_FOWARD(b) ▷ b es una lista de sílabas
  $condition \leftarrow TRUE$
  $k \leftarrow 1$
  $P_0 \leftarrow$ PROBABILITY(b)
  $split_b \leftarrow []$
  while condition = TRUE **and** len(b) >k **do**
    $A_c, A_r \leftarrow b[0:k], b[k:]$
    $B_c, B_r \leftarrow b[0:(k+1)], b[(k+1):]$
    $P_A \leftarrow$ PROBABILITY$(A_c) \times$ PROBABILITY(A_r)
    $P_B \leftarrow$ PROBABILITY$(B_c) \times$ PROBABILITY(B_r)
    $condition \leftarrow P_B < P_A$
    if condition = FALSE **and** $P_B > P_0$ **then**
      ADD$(split_b, B_c)$
    else if condition = FALSE **and** $P_B < P_0$ **then**
      ADD$(split_b, b)$
    $k \leftarrow k + 1$
  return $split_b$

Note que los algoritmos requieren un cálculo de probabilidad, probabilidad que calcula con base en las frecuencias relativas de las palabras si la palabra es conocida y usando (2.12) si la palabra es desconocida. Además, la probabilidad se requiere que los bloques estén concatenados previamente, lo cual se omite dentro de los algoritmos para facilitar su lectura. Por otro lado, las funciones MIX y OPTIMIZE involucradas en el Alg. 2 no se describen a detalle porque, en el caso de la primera consiste en generar las combinaciones que se generan de por medio de los algoritmos de segmentación y, la segunda consiste en seleccionar aquella combinación con probabilidad mayor. La razón es también que sea sencilla la lectura.

3.4 Pre-procesamiento de la locuciones

Para ilustrar la aplicación de los métodos discutidos en el capítulo anterior se tomará de ejemplo algunas de las publicaciones obtenidas para una palabra específica.

Algoritmo 4 Algoritmo de Segmentación hacia atras

```
1: procedure SEGMENTATION_FOWARD(b)                    ▷ b es una lista de sílabas
2:     condition ← TRUE
3:     k ← n − 1                                       ▷ n es el número de elementos de b
4:     P_0 ← PROBABILITY(b)
5:     split_b ← []
6:     while condition = TRUE and 0 >k do
7:         A_c, A_r ← b[k :], b[: k]
8:         B_c, B_r ← b[(k − 1) :], b[: (k − 1)]
9:         P_A ← PROBABILITY(A_c) × PROBABILITY(A_r)
10:        P_B ← PROBABILITY(B_c) × PROBABILITY(B_r)
11:        condition ← P_B < P_A
12:        if condition = FALSE and P_B > P_0 then
13:            ADD(split_b, B_c)
14:        else if condition = FALSE and P_B < P_0 then
15:            ADD(split_b, b)
16:        k ← k − 1
17:    return split_b
```

Texto bruto. La Fig. 3.2 muestra como se ven algunos de publicaciones obtenidas[6], donde cada publicación se encuentra separada por una cadena compuesta por exactamente 10 símbolos de porcentaje.

Separación. La separación consiste en en segmentar el texto completo en una lista de elementos donde cada elemento sea una publicación, luego separar las en palabras tentativas cada publicación. A continuación se muestran los *tweets* de la Fig 3.2 después de la separación donde se resaltan los términos problemáticas en rojo[7]; mientras que en azul se resultan los términos poco frecuentes:

> Instalan Isotanque de oxígeno de 12 mil litros para atender a pacientes de COVID 19 en el Hospital Centinela Rosario Vera Zurita ubicado en el municipio Gran Sabana del estado Bolívar informó el gobernador @GobJustoNoguera #RadicalesContraLaCovid19
>
> Si defiendes el mal manejo del gobierno de Vizcarra y Sagasti en el tema del covid espero que no te des cuenta de que te equivocaste cuando busques atención médica sin resultado Es fácil llenarse la boca de tonteras hasta que no encuentras cama ni oxígeno
>
> Clase obrera de PDVSA y Sidor logró el primer envío de 10 280 Kg de oxígeno líquido al Área Metropolitana de Caracas para la batalla contra la COVID 19 #CuidémonosDelVirus

[6]Por cada palabra se generaron 10 archivos, y no existe una justificación técnica al respecto de porque se prefirió generar 10 archivos por palabra en lugar de uno. Esto obedeció a que se prefirió tener algunos ejemplos de todas las palabras para comenzar a explorar.

[7]Pudiera parecer sorprendente que términos como *psvsa* hayan sido reconocidos, pero en efecto forman parte del CREA.

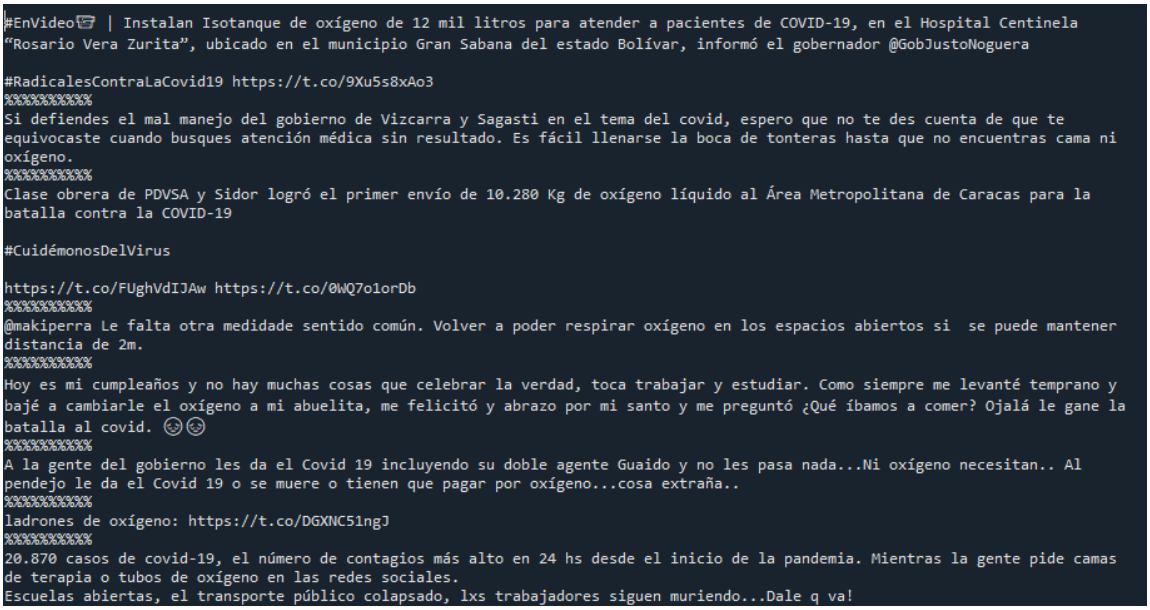


Figura 3.2: Imagen que muestra algunas de las publicaciones en bruto sobre la palabra "oxígeno". Se muestra en imagen porque algunos caracteres no son manejables en LaTeX.

> @makiperra Le falta otra medidade sentido común Volver a poder respirar oxígeno en los espacios abiertos si se puede mantener distancia de 2m
>
> Hoy es mi cumpleaños y no hay muchas cosas que celebrar la verdad toca trabajar y estudiar Como siempre me levanté temprano y bajé a cambiarle el oxígeno a mi abuelita me felicitó y abrazo por mi santo y me preguntó Qué íbamos a comer Ojalá le gane la batalla al covid
>
> A la gente del gobierno les da el Covid 19 incluyendo su doble agente Guaido y no les pasa nada Ni oxígeno necesitan Al pendejo le da el Covid 19 o se muere o tienen que pagar por oxígeno cosa extraña
>
> ladrones de oxígeno
>
> 20 870 casos de covid 19 el número de contagios más alto en 24 hs desde el inicio de la pandemia Mientras la gente pide camas de terapia o tubos de oxígeno en las redes sociales Escuelas abiertas el transporte público colapsado lxs trabajadores siguen muriendo Dale q va
>
> @canarias7 Sanidad los expertos que no existen y la madre que los parió a todos El oxígeno gratuito y es de todos quienes se creen dueños para permitir si respiramos o no en nuestro descanso y más en la playa o el campo
>
> @abc es Desobediencia civil y no hacer caso de un gobierno corrupto con conflicto de intereses en la venta de mascarillas No al bozal mi vida mi oxígeno Prohibición de mascarilla al aire libre ya

Segmentación. La segmentación emplea los algoritmos descritos en la sección 3.3. La aplicación lleva a obtener lo siguiente:

Instalan Isotanque de oxígeno de 12 mil litros para atender a pacientes de COVID19 en el Hospital Centinela Rosario Vera Zurita ubicado en el municipio Gran Sabana del estado Bolívar informó el gobernador @GobJustoNoguera radicales contra la covid19

Si defiendes el mal manejo del gobierno de Vizcarra y Sagasti en el tema del covid espero que no te des cuenta de que te equivocaste cuando busques atención médica sin resultado Es fácil llenarse la boca de tonteras hasta que no encuentras cama ni oxígeno

Clase obrera de PDVSA y Sidor logró el primer envío de 10280 Kg de oxígeno líquido al Área Metropolitana de Caracas para la batalla contra la COVID19 cuidémonos del virus

@makiperra Le falta otra medidade sentido común Volver a poder respirar oxígeno en los espacios abiertos si se puede mantener distancia de 2m

Hoy es mi cumpleaños y no hay muchas cosas que celebrar la verdad toca trabajar y estudiar Como siempre me levanté temprano y bajé a cambiarle el oxígeno a mi abuelita me felicitó y abrazo por mi santo y me preguntó Qué íbamos a comer Ojalá le gane la batalla al covid

A la gente del gobierno les da el Covid19 incluyendo su doble agente Guaido y no les pasa nada Ni oxígeno necesitan Al pendejo le da el Covid19 o se muere o tienen que pagar por oxígeno cosa extraña

ladrones de oxígeno

20870 casos de covid19 el número de contagios más alto en 24 hs desde el inicio de la pandemia Mientras la gente pide camas de terapia o tubos de oxígeno en las redes sociales Escuelas abiertas el transporte público colapsado lxs trabajadores siguen muriendo Dale q va

@canarias7 Sanidad los expertos que no existen y la madre que los parió a todos El oxígeno gratuito y es de todos quienes se creen dueños para permitir si respiramos o no en nuestro descanso y más en la playa o el campo

@abc es Desobediencia civil y no hacer caso de un gobierno corrupto con conflicto de intereses en la venta de mascarillas No al bozal mi vida mi oxígeno Prohibición de mascarilla al aire libre ya

PoS y NER. Las tareas de PoS y NER fueron realizadas usando el Stanford CoreNLP *v4.2*. Los modelos desarrollados por la Universidad de Stanford pueden usarse para realizar un pre-procesamiento completo en inglés y ofrece soporte parcial de algunas tareas para Árabe, Chino, Fránces, Alemán y Español (Manning et al., 2014).

Lematizado.

3.5 Estimación del modelo

CAPÍTULO 4

Resultados

En esta sección se presentan varios resultados y la mayoría pueden explorarse de manera casi independiente uno de otro. Sin embargo, todos los resultados presentados en la tesis, salvo los resultados presentados en la sección 4.1, están reducidos a las 40,000 palabras de uso más frecuente con base en el CREA.

4.1 Corpus

4.2 Análisis de las proyecciones

El *embedding* de palabras es la representación de N palabras en un subespacio de $\mathbb{R}^K$ de dimensión entre 100 y 600. Bajo la métrica coseno, por ejemplo, la palabra *madre* está altamente relacionada[1] con la palabras: *padre*, *hija*, *mamá*, *hermana* y *abuela*.

El ejemplo es un excelente ejemplo de como la métrica coseno recupera relacionadas. Pero es necesario hacer notar que la métrica no establece una relación de orden adecuada. Mayor cercanía debería traducirse en mayor parecido en significado. Pero, en el ejemplo mencionado se tiene que la distancia de *madre* con *padre* fue de 0.76, de *madre* con *hija* fue de 0.81, de *madre* con *mamá* fue de 0.78, de *madre* con *hermana* fue de 0.77 y, de *madre* con *abuela* de 0.75. Por lo tanto, la palabra *hija* está más relacionada con la palabra *madre madre* que la palabra *mamá*.

Otro detalle particular radica en determinar un criterio para determinar "alta" similaridad. Al relajar el criterio de similaridad la palabra *madre* se encuentra relacionada con varias palabras como *mujer*, *familia*, *marido*, *nuera*, *tía*. Palabras que con claridad tienen una relación con la palabra *madre*. Esto permite establecer un campo semántico donde existen todo tipo de relaciones: Meronomia, donde por ejemplo, *madre* e *hija* forman parte de *familia*; Sinonimia en la relación de *madre* con *mamá*; Hiponimia en la relación *abuela* y *madre* y; Lineales en la relación *madre* e *hija*.

Sin embargo, bajo ese mismo umbral de similaridad, la palabra *vida* resulta similar a *cotidiana*. Lo cual da sentido pues "vida cotidiana" es un compuesto común; pero no rescata la misma información que se rescata de la palabra *madre* pues *cotidiana* no forma parte del campo semántico de *vida* por si misma; aunque *vida cotidiana* sí podría considerarse parte del campo semántico de *vida*[2].

Entonces que significan las p

[1] En este ejemplo, se empleó un corte arbitrario de 0.75 para determinar esto. En la literatura no autor que especifique donde se debe hacer el corte para recuperar las palabras "similares".

[2] Los ejemplos mencionados son producto de un análisis exhaustivo de la información contenida `distance_angle.np` y requiere al menos 13 GB de RAM dispoible para su carga completa.

4.3 Propuestas en pre-procesamiento del español

4.4 Análisis de métricas

La metodología descrita en el capítulo 3 permite obtener

4.5 Aplicaciones

Dentro del PLN existen tareas que podrían llamarse nucleares como lo es la modelación del lenguaje, el cual es el tema principal de este trabajo, y tareas que podrían llamarse como finales o aplicaciones que son la parte visible del PLN. De acuerdo con Srinivasa-Desikan (2018) dentro de estas aplicaciones están los asistentes virtuales, las herramientas de atención al cliente como los chatbots, el análisis de sentimiento y la clasificación de documentos. En esta última parte de los resultados se presentan algunas aplicaciones desarrolladas soportadas en parte de los resultados obtenidos previamente mencionados: Bot Agricultor, Analizador de Opiniones de Foros Educativos y Facebook Miner[3].

4.5.1 Bot Agricultor

A través del Sistema Nacional del Información e Integración de Mercados (SNIIM) la Secretaría de Economía proporciona información histórica sobre tres mercados importantes en el país: agrícola, pecuario y pesquero. En 2018 Detags y la Universidad Autónoma de Querétaro acercaron la información del mercado agrícola al pequeño y mediano agricultor con un resultado favorable por medio de una plataforma de Shiny R y de la experiencia obtenida surgieron dos peticiones: precios futuros y consultas sencillas e informativas (Robles, 2018). El bot agricultor representa una solución a la petición de consultas sencillas[4].

El bot agricultor es un chatbot. Un chatbot, de acuerdo con Khan y Das (2017) es un programa que procesa lenguaje natural del usuario y genera una respuesta útil al usuario; aunque en la práctica la mayoría sigue una estructura basada en reglas donde se le va pidiendo al usuario que responda una pregunta con alguna de las opciones válidas mostradas en pantalla para al final dar respuesta a la solicitud (Fig 4.1).

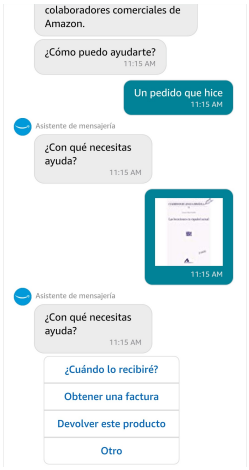
En el contexto de las consultas al SNIIM se definieron cuatro tipos de peticiones que fueron clasificadas en: resumen, predicción, precios (históricos) y comparación, las cuales se implementaron en un chatbot de Telegram similar al empleado por Sci-Hub[5]. Tales peticiones tienen una estructura básica de solicitud:

- `@Resumen Aguacate Hass en Querétaro`
- `@Precio Aguacate Hass en Queretaro 17-04-2020`
- `@Predice Aguacate Hass en Queretaro 7 días`

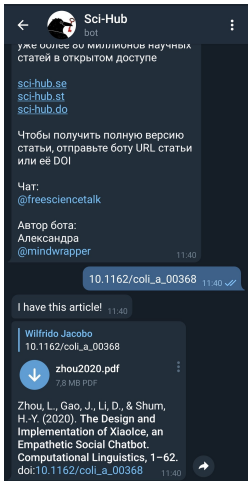
[3]Cada una de las aplicaciones surgió de las pláticas con otras personas que me contaron una idea y que en su momento me pareció una buena para poner a prueba parte de los resultados obtenidos en este trabajo.

[4]A diferencia de código fuente usado a lo largo de este trabajo, el Bot Agricultor se encuentra en el siguiente repositorio: `https://github.com/Wilfridovich17/germinando`.

[5]Para más información para la creación de un bot en Telegram se puede consultar la siguiente liga `https://core.telegram.org/bots`

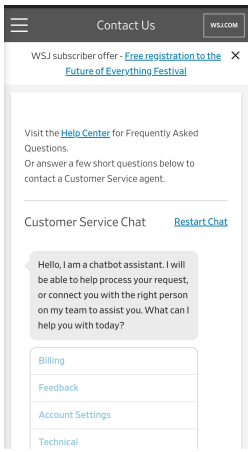


(a) Chatbot de Amazon



(b) Chatbot del Sci-Hub

(c) Chatbot de Telcel



(d) Chatbot del Wall Street Journal

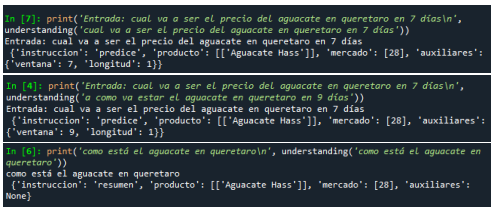
Figura 4.1: Ejemplos de chatbots. Obtenidos mediante una captura del pantalla al sitio del proveedor del servicio al interactuar con el chatbot.

- `@Compara Aguacate Hass en Queretaro 40`

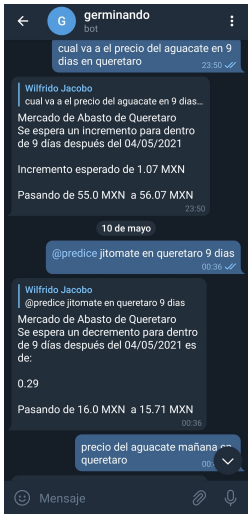
Estas instrucciones se traducen de manera sencilla a una *orden* la cual se maneja usando un diccionario de Python. Por ejemplo la instrucción *predice* se codifica como sigue:

```
orden = {'instruccion':'resumen', 'producto':[['Aguacate Hass']],
    'mercado':[[28]], 'auxiliares':{'Ventana':7, 'Longitud':1}}
```

Las solicitudes con estructura básica fueron creadas para que quien interactúe con el chatbot conozca de antemano qué está pidiendo y qué va a terminar entendiendo el chatbot. Sin embargo, se agregó al chatbot un módulo de lenguaje natural, módulo que le permite transformar instrucciones *más naturales* en la estructura de orden antes mencionada (Fig. 4.2a). Mientras que, a nivel usuario, las solicitudes son procesadas como se muestra en la Fig. 4.2b.



(a) Ejemplo del funcionamiento del módulo de procesamiento natural del bot agricultor.



(b) Ejemplo del funcionamiento del bot agricultor a nivel usuario.

Figura 4.2: Bot agricultor donde se muestra la parte trasera del desarrollo y la parte frontal del desarrollo.

4.5.2 Facebook Miner

4.5.3 Análisis de opinión en foros educativos

De acuerdo con Näslund-Hadley (2020) para el Banco de Desarrollo Inter-Americano, en la educación a distancia ocasionada por la crisis provocada por el SARS-CoV-2, los profesores sienten que deben estar 24/7 para sus alumnos y dicen sentirse abrumados debido a que reciben una enorme cantidad de llamadas y mensajes fuera de su horario laboral. Además, Allen et al. (2020) mencionan que la carga laboral de los docentes aumentó drásticamente debido a la generación y adecuación del material para su distribución vía web así como el registro completo de todas las actividades. Sin embargo, el incremento de la carga docente es resultado de las medidas para disminuir la deserción escolar al mantener actividades y contacto con los estudiantes en un horario extendido.

Algunas escuelas dentro de sus medidas para mantener la interacción, han optado por crear foros de discusión para las materias de los estudiantes y fomentar la reflexión e intercambio de ideas entre los estudiantes. Sin embargo, a diferencia del aula tradicional donde la retroalimentación y moderación del docente se realizaba al momento, disponibilidad de los foros impide que los docentes intervengan de manera fluida. En ese sentido, el PLN puede ofrecer una solución parcial mediante una adecuación de un análisis de opinión semántico con el fin de corregir el rumbo de estudiantes que han llegado a una idea que pueda clasificarse de alguna manera como contradictoria.

CAPÍTULO 5

Conclusiones

APÉNDICE A

Diccionario de términos lingüisticos

Bibliografía

Allen, J., Rowan, L., y Singh, P. 2020. Teaching and teacher education in the time of covid-19. *Asia-Pacific Journal of Teacher Education*, 48(3):233–236.

Allerton, D. J. 1979. *Essentials of Grammatical Theory. A Consensus View of Syntax and Morphology*. ROUTLEDGE.

Bellegarda, J. R. y Monz, C. 2016. State of the art in statistical methods for language and speech processing. *Computer Speech & Language*, 35:163–184.

Billal, B., Fonseca, A., y Sadat, F. 2016. Efficient natural language pre-processing for analyzing large data sets. En *2016 IEEE International Conference on Big Data (Big Data)*, pp. 3864–3871. IEEE.

Brants, T., Popat, A. C., Xu, P., Och, F. J., y Dean, J. 2007. Large language models in machine translation. En *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pp. 858–867, Prague, Czech Republic. Association for Computational Linguistics.

Bruce, P., Bruce, A., y Gedeck, P. 2020. *Practical Statistics for Data Scientists: 50+ Essential Concepts Using R and Python*. O'Reilly Media.

Cano-Felix, O. 2019. Identificación de unidades léxicas y sintácticas de texto informal en español. Tesis de maestría, Universidad Autónoma de Querétaro.

Chapagain, A. 2019. *Hands-On Web Scraping with Python: Perform advanced scraping operations using various Python libraries and tools such as Selenium, Regex, and others*. Packt.

Chomsky, N. 1956. Three models for the description of language. *IRE Transactions on information theory*, 2(3):113–124.

Chomsky, N. 1974. *Estructuras sintácticas*. Siglo xxi.

Chomsky, N. 1977. *Problemas actuales en teoría lingüística: temas teóricos de gramática generativa*. Siglo xxi.

Clark, E. y Araki, K. 2011. Text normalization in social media: progress, problems and applications for a pre-processing system of casual english. *Procedia-Social and Behavioral Sciences*, 27:2–11.

Cohen, D. 1991. *Introduction to computer theory*, vol. 2. Wiley New York.

Damerau, F. J. 1964. A technique for computer detection and correction of spelling errors. *Communications of the ACM*, 7(3):171–176.

Flouri, T., Giaquinta, E., Kobert, K., y Ukkonen, E. 2015. Longest common substrings with k mismatches. *Information Processing Letters*, 115(6-8):643–647.

Fox, C. 1989. A stop list for general text. En *Acm sigir forum*, vol. 24, pp. 19–21. ACM New York, NY, USA.

Gai, R. L., Gao, F., Duan, L. M., Sun, X. H., y Li, H. Z. 2014. Bidirectional maximal matching word segmentation algorithm with rules. En *Advanced Materials Research*, vol. 926, pp. 3368–3372. Trans Tech Publ.

genediazjr 2016. Spanish stopwords collection. `https://github.com/stopwords-iso/stopwords-es`. Accesado: 2018-04-12.

Goldberg, Y. y Levy, O. 2014. word2vec explained: deriving mikolov et al.'s negative-sampling word-embedding method. *arXiv preprint arXiv:1402.3722*.

Goyvaerts, J. y Levithan, S. 2012. *Regular Expressions Cookbook: Detailed Solutions in Eight Programming Languages*. O'Reilly Media.

hammon38 2021. Tweepy. `https://www.tweepy.org/`. Accesado: 2021-01-14.

Harris, Z. S. 1954. Distributional structure. *Word*, 10(2-3):146–162.

Hassan, H. y Menezes, A. 2013. Social text normalization using contextual graph random walks. En *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1577–1586.

Hernández-Figueroa, Z., Carreras-Riudavets, F. J., y Rodríguez-Rodríguez, G. 2013. Automatic syllabification for spanish using lemmatization and derivation to solve the prefix's prominence issue. *Expert systems with applications*, 40(17):7122–7131.

Hládek, D., Staš, J., y Pleva, M. 2020. Survey of automatic spelling correction. *Electronics*, 9(10):1670.

Hualde, J. I. 2013. *Los sonidos del español: Spanish Language edition*. Cambridge University Press.

Khan, R. y Das, A. 2017. *Build Better Chatbots: A Complete Guide to Getting Started with Chatbots*. Apress.

Levenshtein, V. I. 1966. Binary codes capable of correcting deletions, insertions, and reversals. En *Soviet physics doklady*, vol. 10, pp. 707–710. Soviet Union.

López, F. y Romero, V. 2014. *Mastering Python Regular Expressions*. Packt Publishing Ltd.

Manning, C. D., Surdeanu, M., Bauer, J., Finkel, J. R., Bethard, S., y McClosky, D. 2014. The stanford corenlp natural language processing toolkit. En *Proceedings of 52nd annual meeting of the association for computational linguistics: system demonstrations*, pp. 55–60.

Melero, M., Ruiz Costa-Jussà, M., Lambert, P., y Quixal, M. 2016. Selection of correction candidates for the normalization of spanish user generated content. *Natural language engineering*, 22(01):135–161.

Mikolov, T., Chen, K., Corrado, G., y Dean, J. 2013a. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G., y Dean, J. 2013b. Distributed representations of words and phrases and their compositionality. *arXiv preprint arXiv:1310.4546*.

Morin, F. y Bengio, Y. 2005. Hierarchical probabilistic neural network language model. En *Aistats*, vol. 5, pp. 246–252. Citeseer.

Munkres, J. R. 2000. *Topology*. Prentice-Hall.

Norvig, P. 2009. Natural language corpus data. En Segaran, T. y Hammerbacher, J., eds., *Beautiful Data*, cap. 14, pp. 219–242. O'Reilly, EEUU.

Nouvel, D., Ehrmann, M., y Rosset, S. 2016. *Named entities for computational linguistics*. Wiley Online Library.

Näslund-Hadley, E. 2020. 3000 profesores comparten sus experiencias de aprendizaje remoto covid-19. `https://blogs.iadb.org/educacion/es/docentescovid/`. Accesado: 2021-03-01.

Orduña López, J. L. 2011. Estudio gramatical de las locuciones verbales con doble pronombre clítico. *RLA. Revista de lingüística teórica y aplicada*, 49(2):87–110.

Otero, C.-P. 1975. Terminología y teoría gramatical. *Verba 2*, pp. 13–38.

Pennington, J., Socher, R., y Manning, C. D. 2014. Glove: Global vectors for word representation. En *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pp. 1532–1543.

Real Academia Española: Banco de datos (CREA) 2008. Corpus de referencia del español acutal. `http://www.rae.es`. Accesado: 2021-01-14.

Robles, B. 2018. Aplicación sobre precios agrícolas se expande al país. `https://www.debate.com.mx/losmochis/detag-aplicacion-monitoreo-de-precios-mercados-agricolas-20180225-0092.html`. Accesado: 2021-05-12.

Rojas, R. 1996. The backpropagation algorithm. En *Neural networks*, pp. 149–182. Springer.

Rong, X. 2014. word2vec parameter learning explained. *arXiv preprint arXiv:1411.2738*.

Ross, S. M. 2014. *Introduction to probability models*. Academic press.

Sahlgren, M. 2008. The distributional hypothesis. *Italian Journal of Disability Studies*, 20:33–53.

Shi, T. y Liu, Z. 2014. Linking glove with word2vec. *arXiv preprint arXiv:1411.5595*.

Small, T. A. 2011. What the hashtag? a content analysis of canadian politics on twitter. *Information, communication & society*, 14(6):872–895.

Srinivasa-Desikan, B. 2018. *Natural Language Processing and Computational Linguistics: A practical guide to text analysis with Python, Gensim, spaCy, and Keras*. Packt Publishing Ltd.

Sutton, C. y McCallum, A. 2019. An introduction to conditional random fields for relational learning. En Getoor, L. y Taskar, B., eds., *Introduction to Statistical Relational Learning*, cap. 4, pp. 93–128. MIT Press, EEUU.

Twitter, Inc. 2021. Twitter for developers. `https://developer.twitter.com/en`. Accesado: 2017-10-29.

Waggener, B., Waggener, W. N., y Waggener, W. M. 1995. *Pulse code modulation techniques*. Springer Science & Business Media.