BENEMÉRITA UNIVERSIDAD AUTÓNOMA DE PUEBLA
FACULTAD DE CIENCIAS DE LA COMPUTACIÓN

Tesis para obtener el título de:
Licenciatura en Ing. en Ciencias de la Computación

"Captura de atributos discriminativos"

Presenta:
Alberto Tapia Palacios

Asesora:
Dra. Darnes Vilariño Ayala

Puebla – México Enero 2020

Resumen

La tarea conocida como captura de atributos discriminativos es propuesta en SemEval2018, la cual surge a partir de los problemas relacionados con los modelos de similitud de palabras, sin embargo, este problema es investigado en las áreas de semántica con semántica léxica y computación con Procesamiento de Lenguaje Natural, en ambos casos mencionan que los lenguajes naturales no están formalizados y siempre están en constante crecimiento. Una propuesta al problema anterior es diferenciar a los objetos por sus características que lo componen o capturar sus atributos discriminativos.

En esta tesis se presenta la investigación, análisis, comparación y resultados de algunos de los diferentes modelos de similitud existentes, junto con modelo propuesto de clasificación de atributos discriminativos.

Abstract

The task known as capture of discriminative attributes is proposed in SemEval2018, which arises from the problems related to the models of similarity of words, however, this problem is investigated in the areas of semantics with lexical semantics and computation with processing of Natural Language, in both cases mention that natural languages are not formalized and are always in constant growth. A proposal to the previous problem is to differentiate the objects by their features that compose it or capture their discriminative attributes.

This thesis presents the research, analysis, comparison and results of some of the different models of similarity, together with the proposed model of classification of discriminative attributes.

Agradecimientos

Quisiera dar mi más sincero agradecimiento a todas aquellas personas que han apoyado en la realización de este nuevo triunfo. En primera instancia agradezco a mis formadores, personas de gran sabiduría quienes se han esforzado por ayudarme a llegar al punto en el que me encuentro. De igual manera a toda mi familia y amigos por el apoyo y animo incondicional recibido.

Contenido

Lista de Figuras

Lista de Tablas

Capítulo 1 Introducción

1.1

En el siguiente capítulo se expondrá una introducción al tema de captura de atributos discriminativos, empezando con lo que dio origen al problema, la investigación en las diferentes áreas que son semántica y computación, asi como algunos conflictos que tiene hasta hoy en día.

1.2 Introducción

En la actualidad el lenguaje es algo fundamental para poder comunicarse por algún medio, ya que permite el desarrollo de nuevas formas de pensamiento y la adquisición de nuevos conocimientos. Existen varias definiciones otorgadas al término de lenguaje, sin embargo, la mayoría coincide en varios aspectos, por lo que se va a definir como un conjunto finito de símbolos dentro un sistema basado en un código. Podemos distinguir entre dos tipos de lenguajes: los lenguajes naturales (como pueden ser inglés, alemán, español, por mencionar algunos) y lenguajes formales (matemático, lógico, programación, entre otros) (Cortez, Vega, Pariona y Huayna, 2014). La disciplina que estudia el significado de las expresiones lingüísticas es la semántica, sus ramas de investigación son la semántica léxica la cual consiste en investigar todo lo relativo al significa de las palabras y semántica composicional que se centra en descubrir la contribución que aporta la estructura y relaciones sintácticas de expresiones complejas (Escandell, 2007).

El avance de las tecnologías de hoy en día ha impulsado el desarrollo de mejorar la comunicación humano-computadora; además de que gran parte de la información se encuentra de forma digital en diferentes tipos de colecciones de datos, desde foros, blogs, wikis hasta redes sociales. Estas colecciones son inmensas, creciendo exponencialmente día a día gracias al impulso de internet, donde la información en su mayoría se encuentra sin ningún tipo de clasificación, generando que las investigaciones de la compresión y el uso de forma automática de los lenguajes naturales se vea incrementado en los últimos años.

Surgiendo así una disciplina llamada Procesamiento del Lenguaje Natural (PLN) que combina la rama de la lingüística y la informática, con el objetivo de modelar el lenguaje humano desde el punto de vista computacional, comúnmente en forma de datos textuales. El enfoque principal de esta área consiste en crear métodos, técnicas y herramientas computacionales que permitan realizar análisis de información y faciliten la búsqueda y organización de dicha información (Karin, 2013).

Algunas de las aplicaciones que desarrollan el PLN son la recuperación y extracción de la información, traducción automática entre idiomas, sistemas de búsquedas preguntas/respuestas, minería de datos, análisis de sentimientos, generación de resúmenes automáticos por mencionar algunas (Hernández y Gómez, 2013; Cortez et al., 2014); lo que requiere el conocimiento profundo del lenguaje para procesar la información y proporcionar resultados fiables.

De las aplicaciones mencionadas anteriormente, una de las tareas más vitales y por su alto grado de complejidad es la detección de similitud semántica entre pares de sentencias, similitud semántica es la medida que existe entre dos textos, frases o palabras, esta medida se obtiene a través de modelos de PLN, asi que el resultado depende del modelo utilizado y la riqueza del lenguaje natural con el que se esté trabajando, mejorar los resultados de esta tarea ayudará a las demás aplicaciones a mejorar sus respuestas en la toma de decisiones automática.

A lo largo de los últimos años la tarea de similitud semántica ha llevado a cabo diferentes investigaciones con diferentes tipos de resultados, ya que para textos de gran longitud puede entregar resultados aceptables, pero en el caso de similitud entre pares de palabras o textos cortos la información es insuficiente para lograr resultados favorables, puesto que los esquemas, modelos y colecciones de datos de hoy en día están basados en representación por frecuencia o vectores de palabras, así que el modelo de similitud semántica sigue siendo una tarea a investigación por su grado de complejidad.

1.3 Formulación del problema

En el área de Procesamiento del Lenguaje Natural (PLN) existen varios métodos de evaluación, que sirven como una buena herramienta para solucionar problemas de similitud semántica, pero deben usarse con mucho cuidado, ya que las evaluaciones de similitud semántica entre palabras supone que existe una sola noción de similitud, por lo que utilizar algunos de los diferentes modelos existentes mostrará resultados diferentes independientemente del problema a resolver, de igual manera no existe un estándar para la representación de palabras en vectores, lo que tiene como consecuencia que en algunos conjuntos de datos resulte difícil encontrar diferencias significativas. En la tabla 1.1 se muestran los diferentes y más conocidos conjuntos de datos de similitud en PLN, estos fueron construidos a partir de una lista de pares de palabras con anotaciones de juicio de similitud humana, desde entonces no hay un estándar para la creación de nuevos conjuntos de datos, por lo que utilizar uno u otro brindara diferentes resultados, obtenido de wordvectors.org (Faruqui y Dyer, 2014).

NO.	DATASET	PARES DE PALABRAS	REFERENCIA
1	WS-353	353	Finkelstein et. al, 2002
2	WS-353-SIM	203	Agirre et. al, 2009
3	WS-353-REL	252	Agirre et. al, 2009
4	MC-30	30	Miller and Charles, 1991
5	RG-65	65	R and G, 1965
6	Rare-Word	2034	Luong et. al, 2013
7	MEN	3000	Bruni et. al, 2012
8	MTurk-287	287	Radinsky et. al, 2011
9	MTurk-771	771	Halawi and Dror, 2012
10	YP-130	130	Yang and Powers, 2006
11	SimLex-999	999	Hill et. al, 2014
12	Verb-143	143	Baker et. al, 2014
13	SimVerb-3500	3500	Gerz et al., 2016

Tabla 1.1 Conjuntos de datos

Faruqui, Tsvetkov, Rastogi y Dyer (2016) describen algunos de los problemas más comunes con las tareas de tipo de evaluación de similitud para los modelos de vectores de palabras, sugiriendo que el uso de estos modelos sin supervisión puede

conducir a resultados incorrectos, por lo que se requiere especial atención a la hora de evaluación y desarrollo de la tarea.

Los problemas encontrados por Faruqui et al. (2016) son:

Subjetividad de la tarea en la cual mencionan que la noción de similitud de palabras es subjetiva y en ocasiones confusa con la relación. Más detalladamente Batchkarov et al. (2016) describen que uno de los principales problemas es la definición de similitud dentro de los modelos existentes ya que los conjuntos de datos contienen una amplia gama de relaciones semánticas tales como la sinonimia que consiste en la relación de igualdad que hay entre el significado de dos o más palabras, antonimia que es la relación de oposición entre los significados de dos palabras, hiperonimia que es la relación existente entre un hiperónimo con otra palabra cuyo significado engloba otras palabras, co-hiperonimia es la que comparte hiperónimo entre sí, por ejemplo fruta es hiperónimo de manzana, mango y pera, asi como manzana es co-hiperónimo de mango, y a su vez de pera, ver Figura 1.1, meronimia se usa para describir una relación parcial entre dos elementos, por ejemplo dedo es meronimio de mano, porque dedo es parte de una mano. Lo que tiene como consecuencia que algunos modelos no distingan entre similitud y relación entre un par de palabras. Lofi (2015) describe que la similitud semántica mide la distancia percibida entre dos conceptos respecto a su tipo o función, por ejemplo: “caballo” es muy similar a “burro” porque son animales equinos. Por el contrario, la medición de relación muestra la relación que tienen los objetos o entidades entre sí, por ejemplo: "Arnés" está muy relacionado con "caballo" pero no es similar, ya que no comparten tipo o función en común, sin embargo, un arnés se usa a menudo en un caballo.

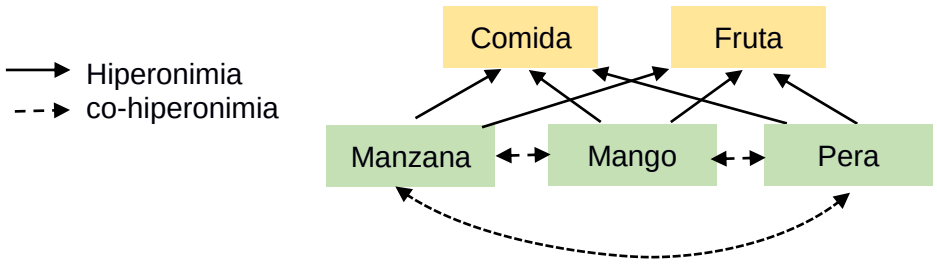


Figura 1.1 Ejemplo de Hiperonimia y co-hiperonimia

Para obtener modelos de aprendizaje generalizables, es necesario asegurarse de que no se sobreajusten a un conjunto de datos determinado. Por lo tanto, los conjuntos de datos suelen dividirse en un conjunto de entrenamiento, desarrollo y prueba en el que el modelo se entrena, ajusta y finalmente se evalúa; los conjuntos de datos de similitud de palabras existentes no están particionados por lo que se recomienda optimizar los vectores de palabras para obtener un mejor rendimiento en el resultado de la tarea (Faruqui et al. 2016).

Otro problema es la incapacidad de descubrir la polisemia es decir que muchas palabras tienen más de un significado en un idioma. Por ejemplo, la palabra “bank” (idioma inglés) puede responder a una institución financiera (banco) o a la tierra cerca de un río (orilla). Sin embargo, en el conjunto de datos WS-353 que se encuentra en la tabla1, el banco recibe una puntuación de similitud de 8.5 / 10 con dinero, lo que resulta que el banco es una institución financiera. Por lo cual se hace una suposición en este sentido y esto puede penalizar un modelo de vector de palabras al capturar un sentido en específico (Faruqui et al. 2016).

El tamaño del conjunto de datos de cada modelo es otro problema que tiene la similitud semántica dentro del PLN, puesto que este varía dependiendo del conjunto de datos que se utilice, porque puede tener más o menos entradas de palabras y afectar directamente a los resultados obtenidos, además de que cada modelo otorga diferentes valores de similitud semántica.

Batchkarov, Kober, Reffin, Weeds y Weir (2016) muestran que la tarea de similitud semántica entre pares de palabras es desafiante dado que el conjunto de datos que se utiliza puede tener una varianza en los resultados, dificultando la diferenciación confiable entre los modelos existentes, por lo que se tiene que tener cuidado con sus limitaciones debido a que la tarea de similitud semántica es una medida aproximada y esta depende de la calidad del modelo.

Tversky (1977) menciona que estas tareas se basan en función de la distancia métrica que representa los objetos como puntos en un espacio de coordenadas a los cuales se asigna un número no negativo llamado distancia de acuerdo con los siguientes axiomas

- Minimidad

La distancia entre dos objetos es igual o mayor que la distancia entre un objeto y sí mismo, lo cual es cero.

$$\delta(A, B) \geq \delta(A, A) = 0 \quad (1)$$

El axioma de la minimalidad implica que la similitud entre un objeto y el mismo es la misma para todos los objetos, sin embargo, no es válida para algunas medidas similares, pues la probabilidad de juzgar dos objetos como iguales en lugar de diferentes no es constante para todos.

- Simetría

  La distancia de *A* hacia *B* es la misma que la distancia de *B* hacia *A*

$$\delta(A, B) = \delta(B, A) \quad (2)$$

  La simetría a veces es vista como una relación simétrica, de la forma *A* es como *B* y de manera direccional *B* es como *A*, pero esto depende de la relevancia relativa de los objetos. Por ejemplo “una elipse es como un círculo, y no un círculo es como una elipse “(Tversky, 1977, p328)

- La desigualdad del triángulo

  La suma de la distancia entre *A* y *B* y la distancia entre *B* y *C* no puede ser menor que la distancia entre *A* y *C*.

$$\delta(A, B) + \delta(B, C) \geq \delta(A, C) \quad (3)$$

  Esto implica que si *A* es bastante similar a *B* y *B* es bastante similar a *C*, entonces *A* y *C* no pueden ser muy diferentes entre sí. Por ejemplo, “Jamaica es similar a Cuba (debido a la proximidad geográfica); Cuba es similar a Rusia (debido a su proximidad política); pero Jamaica y Rusia no son para nada similares.” (como se cita en Tversky, 1977, p329).

“La aplicabilidad de la suposición dimensional es limitada y los axiomas métricos son cuestionables. Específicamente, la minimización es algo problemática,

la simetría es aparentemente falsa y la desigualdad del triángulo no es convincente." (Tversky, 1977).

La diferencia del modelado puede ayudar a capturar aspectos más detallados del significado de las palabras y mejorar el rendimiento del modelo; Tversky (1977) propone que la similitud debe describirse como una comparación de características, más que como una comparación de distancias métricas. Para el dominio de objetos Δ = {a, b, c, ..., n}, donde cada objeto en delta está asociado con un conjunto de características especifico. Las características pueden corresponder a componentes tales como alguna parte del cuerpo, color o peso. La similitud se puede evaluar contando las características que tienen en común dichos objetos.

Krebs y Paperno (2016) proponen la tarea de captura de atributos discriminativos, la cual consiste en capturar las diferencias entre dos conceptos utilizando las características de cada uno; en SemEval 2018 la tarea fue definida junto con su respectivo conjunto de datos (training, validation y test). El objetivo es predecir si una palabra es un atributo discriminativo de otras dos, por ejemplo, dadas las palabras "apple" (manzana) y "banana" (platano), ¿es la palabra "red" (rojo) es un atributo discriminativo de alguna de estas? La diferencia semántica es una relación entre dos conceptos (apple, banana) y una característica discriminativa (red) que caracteriza el primer concepto, pero no el otro. Con lo que se puede decir que la detección de diferencia semántica es una tarea de clasificación binaria: (Apple, banana, red), al determinar si ejemplifica una diferencia semántica o no.

1.4 Objetivo de esta tesis

1.4.1 Objetivos general

Proponer un modelo de atributos discriminativos entre pares de palabras y un atributo extrayendo las diferencias semánticas de cada una.

1.4.2 Objetivos específicos

- Determinar el algoritmo de expansión de cada palabra que componen cada sentencia.

- Proponer un modelo que use el algoritmo anterior.
- Probar el modelo desarrollado.

1.5 Organización de la Tesis

Esta tesis se ha estructurado en los siguientes capítulos principales:

El primer capítulo es una introducción al tema de captura de atributos discriminativos, dando a conocer los orígenes del problema, en el que se presenta la investigación que se encarga del estudio y resolución de este problema dentro de las áreas de semántica en semántica léxica y computacional en PLN respectivamente, asi como algunos conflictos a nivel semántico como es la polisemia y a nivel computacional como son los diferentes modelos de similitud semántica que siguen surgiendo hasta el día de hoy, en el siguiente capítulo se describe a detalle los conflictos mencionados.

En el segundo capítulo se presenta la descripción de los conceptos más relevantes en el área de lingüística del tema de semántica y semántica léxica, así como las relaciones semánticas que la componen, modelos de similitud en el área de la lingüística y fórmulas de similitud semánticas que son aplicadas por algunos modelos de similitud semántica dentro del área computacional.

En el tercer capítulo se hace una revisión bibliográfica sobre el estado actual de las investigaciones sobre las herramientas, tecnologías y conjuntos de datos sobre similitud entre pares de palabras, atributos discriminativos y trabajos relacionados.

El cuarto capítulo describe la tarea de atributos discriminativos, su conjunto de datos, herramientas utilizadas y explicación detallada de las implementaciones del funcionamiento del modelo propuesto.

El quinto capítulo se presenta la evaluación completa y los resultados del modelo propuesto, así como los ajustes que fueron realizados para obtener los mejores resultados.

Al final se encuentran los resultados y la bibliografía de los trabajos que fueron consultados como referencia para el desarrollo de la tesis.

Capítulo 2 Marco teórico

En este capítulo se presenta la descripción de los conceptos más relevantes en el área de semántica, semántica léxica y computación, así como las relaciones semánticas, modelos de similitud en el área de la lingüística y fórmulas de similitud semánticas que son usadas por algunos modelos.

2.1 Lenguaje

Un lenguaje se puede definir como una función que expresa pensamientos y comunicaciones entre la gente, esto se puede realizar mediante signos escritos o señales auditivas (Cortez et al. 2014).

Entre los diferentes tipos de lenguajes existen dos básicos, que son los lenguajes naturales, los cuales se caracterizan por ser utilizados para comunicarse por medio de una colección finita de datos, es decir son caracteres o letras, que en conjunto forman palabras, esta combinación de palabras tiene un significado dentro de un diccionario del lenguaje natural (puede ser inglés, español, francés, alemán por mencionar algunos) para lograr una comunicación efectiva, además de que no tienen el control de ninguna teoría. Por otro lado, están los lenguajes formales (como es el caso de las matemáticas y la lógica) que fueron desarrollados a través del establecimiento de una teoría, la cual presenta las bases de dicho lenguaje. Definiendo así que las lenguas o lenguajes son sistemas complejos, que asocian contenidos de pensamiento y significado a manifestaciones simbólicas de forma escrita, estas secuencias de símbolos son tomadas de una colección ya existente (Polanco, 2000; Cortez et al. 2014).

Las propiedades del lenguaje formal:

- Se desarrollan a partir de una teoría establecida.
- Tienen un componente semántico mínimo.
- La sintaxis elimina expresiones no ambiguas, proporcionando un significado que cualquiera puede comprender.
- Completa formalización del lenguaje.

Las propiedades del lenguaje natural:

- Son desarrollados por enriquecimiento progresivo, es decir que su diccionario crece sin el intento de formación de una teoría.
- La importancia de su carácter progresivo, se debe grandemente a la riqueza de su componente semántico, lo cual depende en gran medida de la lengua y área geográfica donde se encuentre.
- Dificulta una formalización completa, ya que no es lenguaje con reglas bien definidas.

Para el caso de cualquier lenguaje natural su colección finita es el conjunto de letras del alfabeto (caracteres) y símbolos (guión, tilde, apóstrofe u otros) que se utilizan para formar las palabras, por lo tanto, es necesario que estas combinaciones sean correctas y tengan un sentido dentro del lenguaje natural. Las palabras en una oración poseen un significado y tienen su significante, esto quiere decir, que independientemente del significado de cada palabra, el sentido correcto que estas adquieren varía según el contexto en el que se expresen (Polanco, 2000).

2.2 Semántica

La semántica es la disciplina que estudia el significado de las expresiones lingüísticas, palabras, oraciones y textos de una lengua. Comúnmente se separa el estudio del significado de las palabras con la semántica léxica y el estudio de expresiones más complejas (oraciones) con la semántica composicional, la cual parte de que las palabras tienen significado y se centra en descubrir cuál es la contribución que la estructura y las relaciones sintácticas aportan a la construcción de las expresiones complejas (Escandell, 2007). La semántica léxica es el estudio de cómo y qué denotan las palabras de una lengua, en otras palabras, se ocupa del estudio del significado léxico de las palabras (contenido), en oposición al sentido de las palabras gramaticales, es decir que está más enfocada a las categorías abiertas de los sustantivos, verbos y adjetivos, dando por hecho que existe una colección de lista de palabras de una lengua (como se cita en Duta, 2009, p.1).

Escandell (2007) menciona los objetivos de la semántica léxica

- Caracterizar el significado de las palabras, cualquier modelo de descripción semántica debe caracterizar en que consiste el significado de las palabras y en qué términos puede describirse, una de las maneras consiste en hacer una colección de las palabras de una lengua e indicar su significado, tal es el caso de los diccionarios que son los instrumentos de descripción del significado lingüístico.
- Relaciones que mantienen las palabras en virtud de su significado con otras palabras.
- Definir los diferentes tipos de significados de una palabra, pues la diferencia de significado entre un par de palabras a veces no tiene que ver con el tipo de entidad referida, tal es el caso de las palabras fresa y frutilla el cual depende de la procedencia geográfica, que incluye parámetros de variación del significado.
- Explicar la variación contextual del significado de las palabras según la situación, pues este significado varía según la función del contexto lingüístico en que se ocupe, por ejemplo, para los verbos el significado puede variar según sea su uso, como abrir significa cosas un poco diferentes, si abre una botella, un libro o la sesión de algún evento.
- Surgimiento de nuevos significados, debido a la flexibilidad de adaptación, por ejemplo, en años posteriores la palabra ratón se refería a un animal de tipo roedor, sin embargo, hoy en día resulta más familiar con un dispositivo informático.

Algunas dificultades con una teoría del significado léxico son los problemas con los instrumentos empleados en la descripción, como son las definiciones de las palabras, ya que estas se proporcionan bajo la misma lengua que se utilice, así que para definir una palabra hay que definir también el significado de las palabras empleadas y así sucesivamente, produciendo un inmediato efecto de circularidad, en el que para definir la lengua objeto de estudio se tiene que emplear de la misma manera para describir y caracterizar una entidad de la realidad (Escandell, 2007).

Problema de la naturaleza del significado.

- Distinción entre conocimiento léxico y enciclopédico, es decir, reconocer el conocimiento que se conoce (adquirido por una persona) sobre la realidad designada por las palabras (definición de un diccionario de la lengua).
- Lexicalización de los conceptos, las palabras imponen una estructura a la realidad y al modo de percibirla, esto varia notablemente de una lengua a otra.
- Problema de variación contextual del significado, puesto que la carga del significado que puede adquirir una palabra está en función de muchos factores extralingüísticos.

2.2.1 El metalenguaje de la semántica léxica

Un metalenguaje es un lenguaje que se usa para hablar de aspectos propios de una lengua para describirla, enunciar o analizar el lenguaje de las propiedades exigibles de un tipo de vocabulario, con el fin de ayudar a situar la frontera entre conocimiento lingüístico y conocimiento enciclopédico.

Algunas de las unidades básicas descritas por Escandell (2007) son:

Una palabra que es una cadena de letras que aparece entre dos espacios, sin embargo, no constituye una caracterización general, como lo definen varios autores una sola definición de palabra resulta problemática cuando se quiere aplicar a los artículos, preposiciones o las conjunciones ya que estas son más complejas, además de que una palabra puede tener otros componentes, pero estos no pueden reordenarse o admitir la interpolación de otras palabras (Escandell, 2007, p.21).

El lexema es un elemento fundamental de la semántica léxica de un lenguaje, pues es la base de una palabra, es decir la parte que se mantiene igual en todas sus variaciones y derivaciones. El lexema aporta el núcleo conceptual del significado; por ejemplo, para el lexema del lenguaje español libr-, están las formas de libro, libros, libritos, sin embargo, no todas las palabras o variaciones de una lengua aparecen en el diccionario.

Los denominados lema son cada una de las palabras que se definen en un diccionario de la lengua, los criterios básicos para la selección de las palabras variables son en género (por ejemplo, masculino, singular) y para las formas verbales (infinitivo).

Referente es la entidad que alude una expresión, lo que hace un vínculo entre lenguaje y realidad, como es el caso de los nombres propios, que se ligan directamente a las entidades que se refieren, por lo cual los nombres propios pueden tener o no significado, ya que no es un factor relevante para identificar su referente, por ejemplo, el nombre Rosa de una persona no exige tener ninguna de las propiedades de dicha flor. Al excluir los nombres propios del resto de las palabras, la relación con la realidad se establece según su significado, lo que permite referir a entidades del mundo real.

El significado de las palabras actúa como mediador entre la conceptualización y el lenguaje, por lo que una definición de significado en general no puede ser única, ya que tienen perspectivas diferentes, algunos de los diferentes tipos de significados se mencionan a continuación (Duta, 2009; Escandell, 2007).

El significado descriptivo o denotativo es aquel que nos permite identificar la realidad extralingüística a la que dicha expresión se refiere, de naturaleza objetiva e invariable de una situación a otra, por ejemplo, el significado descriptivo para la palabra autobús permite etiquetar ciertos objetos de la realidad bajo el criterio de autobús y diferenciarlos de otros que compartan propiedades en común como la furgoneta. Por otro lado, las palabras pueden tener asociado diferente contenido, que es variable o inestable ya que este corresponde al significado no descriptivo, por ejemplo, la palabra autobús puede estar asociada a las ideas de atasco, aglomeración, retraso, viajes, entre otros, pero ninguna de las ideas anteriormente mencionadas forma parte del significado descriptivo de la palabra. Esto tiene como consecuencia no poder establecer un significado general, pues las dos clases son diferentes, según posean uno u otro tipo de significado (Escandell, 2007).

Existen dos tipos de palabras definidas las palabras llenas como es el caso de sustantivos, verbos o adjetivos y palabras vacías como son conjunciones, artículos, etc. Sin embargo, al decir palabras vacías no implica que no tengan

significado, simplemente es de otra naturaleza más general, ya que no proporciona información específica de una palabra. La diferencia en ambos tipos de significado se categoriza como el significado léxico y significado gramatical. Las palabras que tienen significado léxico remiten a conceptos, a partir de los cuales es posible identificar entidades, propiedades, actividades y estados, mientras que las expresiones gramaticales indican de manera abstracta el modo en que hay que combinar los conceptos (Escandell, 2007).

Según Escandell (2007) las características que identifican las unidades con significado léxico son:

- Forman clases abiertas en las que es posible incorporar fácilmente nuevos miembros.
- Tienen contenido descriptivo que permite identificar tipo de entidades, según el conocimiento enciclopédico.
- Representaciones conceptuales de las palabras.

Las unidades de significado gramatical tienen propiedades opuestas como son:

- Clases cerradas en las que no se incorporan fácilmente nuevos miembros.
- No poseen contenido descriptivo.

2.3 Relaciones semánticas

Uno de los métodos tradicionales para definir una palabra mediante otras es la utilización de los diccionarios, lo que se puede ver como una red que tiene numerosas relaciones léxicas entre palabras y entidades con el mundo real; esto provoca el problema de circularidad como se describió anteriormente, el cual consiste en definir cada una de las palabras que se usan en la definición, y en sus respectivas definiciones también, hasta que finalmente podemos quedarnos sin palabras o reutilizar una de las palabras que se intenta definir (Duta, 2009, p.2).

Eggins (2004) menciona que el análisis de las relaciones léxicas se puede ver como una manera de descripción sistemática, las palabras tienen relación con

otras y así pueden formar nuevos conjuntos léxicos para proporcionar una clasificación organizada de las relaciones léxicas (Duta, 2009, p.2).

Escandell (2007) argumenta que las propiedades más notables del léxico es que los significados entre palabras no son independientes entre sí, ya que están conectados por diferentes tipos de relaciones.

2.3.1 Relación de identidad y semejanza

La relación entre dos palabras que tienen un significado en común es denominada sinonimia, un par de palabras que se pueden intercambiar en cualquier contexto sin que produzca efectos comunicativos u otro aspecto relacionado con el significado.

En algunos casos concretos la sinonimia resulta difícil de decidir si dos palabras son sinónimas, ya que el significado presenta una realidad multifacética, pues esto depende en algunos casos de la diferencia en distribución geográfica.

A continuación, se mostrarán algunos ejemplos, la palabra coche es más utilizada en España y la palabra carro en América, ambas tienen el mismo significado, por lo que sustituir una por otra resultaría completamente indistinto. Por otro lado, para las palabras bajar y descender no tiene diferencias de significados notables, sin embargo, en diferentes contextos, estas palabras se comportan de forma diferente, por ejemplo, podemos decir “la bolsa ha bajado”, pero resulta más extraño decir “la bolsa ha descendido”, sin embargo, son intercambiables, pero “El hombre desciende del mono” es posible remplazar desciende por proviene; pero no por baja (Escandell 2007).

Entonces dos palabras pueden tener significados muy cercanos, pero estos significados no son totalmente idénticos, lo que tiene como consecuencia que no sean intercambiables en cualquier contexto, ya que habrá algo que las diferencie.

Los sinónimos absolutos son dos palabras que tienen el mismo contexto semántico y además se pueden intercambiar en cualquier otro contexto, sin que produzca ninguna modificación en sus efectos comunicativos u otro aspecto con su relación con el significado; sin embargo, el número de estos términos es muy

pequeño, por lo que si se flexibilizan las condiciones se consigue aumentar el número de relaciones de identidad y semejanza (Escandell 2007).

Una solución para el problema de las diferentes dimensiones del significado como lo propone Escandell (2007) es considerar solamente el contenido descriptivo de las palabras, denominando sinónimos a dos términos que poseen el mismo contenido descriptivo, aunque estos no puedan intercambiarse en todos los contextos, por ejemplo, las palabras: vivienda, morada, domicilio, residencia, comparten el mismo contenido descriptivo básico, pero ello no significa que pueden ser utilizadas de manera adecuada en cualquier contexto, esto depende más de la norma y uso que conocemos hasta el día de hoy, la cual consiste en preferir una combinación sobre otra, aunque ambas tengan su contenido descriptivo de manera equivalente. Para las palabras bajar y descender se puede observar que no todos sus significados son equivalentes, pero esto no es un problema para establecer una relación de sinonimia de los significados, puesto que como se mencionó anteriormente, depende del contexto de la situación en la cual se van a utilizar, ver Figura 2.1. Cabe mencionar que los nombres propios carecen de significado, puesto que tienen una identidad de referente, tal es el caso de las expresiones y nombre de: “Nueva York”, “la gran manzana” o “la ciudad de los rascacielos” las cuales hacen referencia a la misma ciudad, por lo tanto, se dice que el ejemplo anterior son expresiones correferenciales.

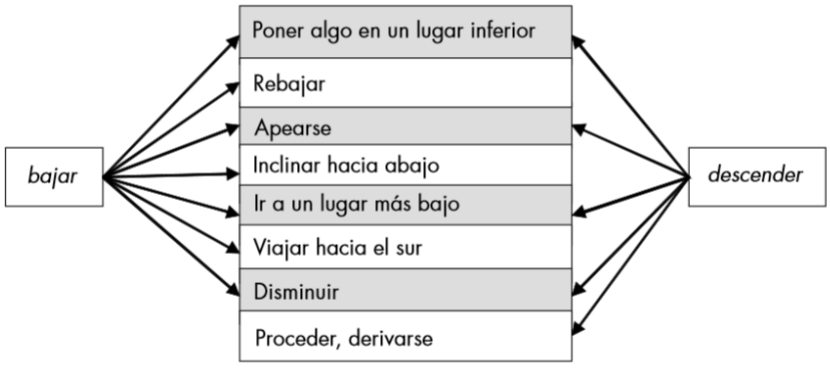


Figura 2.1 Relación de identidad entre significados

2.3.2 Relación de inclusión

Las relaciones de inclusión es cuando el significado de un término está contenido en el significado de otro, por ejemplo, álamo y árbol, esto ejemplifica las relaciones de inclusión “ser un tipo de”, un álamo es un tipo de árbol, siendo así que árbol es el contenido semántico general y álamo contendrá todos sus rasgos, además de aquellos que permite diferenciarlo de otros tipos de árboles. Otro ejemplo está en el significado de la palabra dedo que está necesariamente incluido en el de la palabra mano o pie, ya que un dedo se caracteriza como cada uno de los cinco apéndices articulados en que terminan la mano y el pie, por lo que la relación de inclusión está basada como “parte/todo” (Escandell 2007).

Escandell (2007) clasifica las relaciones de inclusión de la siguiente manera, tomando en cuenta que se consideran las relaciones entre significados y no relaciones entre objetos.

Hiponimia (hipónimo) es la relación que se establece entre el significado de un término más reducido y otro significado más amplio que queda incluido, es decir el significado del término más general es una parte del significado del término más específico.

Hiperonimia (hiperónimo) es la relación inversa de la hiponimia, donde del significado más general se derivan diferentes sub especificaciones.

Co-hiponimia (co-hipónimos) es la relación que se establece entre los significados que comparten el mismo elemento en común (hiperónimo).

Meronimia es la relación de inclusión entre significados que depende de la relación “parte/todo”, es decir la conexión entre el significado de una palabra que indica una parte correspondiente de un todo.

Co-meronimia es la relación que se establece entre los significados que comparten una totalidad en común.

Merónimo al término incluido en la relación de meronimia (parte)

Holónimo al término que incluye (todo)

Retomando el ejemplo de álamo y árbol anteriormente mencionado se le llama hipónimo a cada uno de los términos de significado más específico (álamo,

roble, pino, olivo, encino), hiperónimo al término de significado más general (árbol) y al conjunto de hipónimos se le llama co-hipónimos, ver Figura 2.2. Por lo que se puede decir que desde el punto de vista del significado que el hipónimo posee un significado más general y este también incluye al hiperónimo. (Escandell 2007)

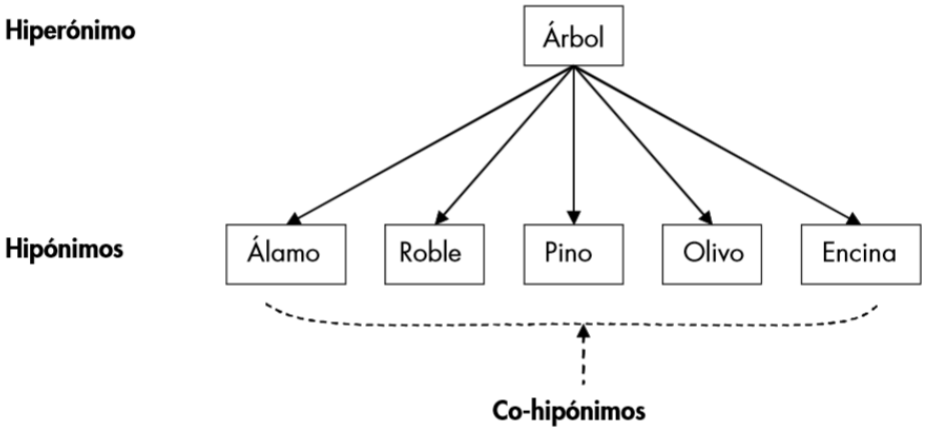


Figura 2.2 Ejemplo hiperónimo-hipónimo

Las relaciones de meronimia resultan ser más complejas por la idea intuitiva de incluir el significado del todo en el de una de sus partes, lo que conlleva a que no existe una única clase de relación ya que tiene varias posibilidades. Un ejemplo de lo anterior es los merónimos de la palabra pie, los cuales representan algunas partes, pero no todas tienen el mismo tipo de cohesión con respecto al conjunto (holónimo), lo que muestra que algunas partes son unidas al conjunto (dedo y pie), mientras que otras partes son integradas al conjunto (planta y pie), ver Figura 2.3. Otro claro ejemplo es el de la palabra bicicleta que tiene que tener como característica dos ruedas, pues si tiene más o tiene menos, ya no sería bicicleta (partes integradas), otra característica es que algunas bicicletas tienen guardafangos, pero no todas, pues no es una parte esencial que defina una bicicleta (parte unidad). (Escandell 2007)

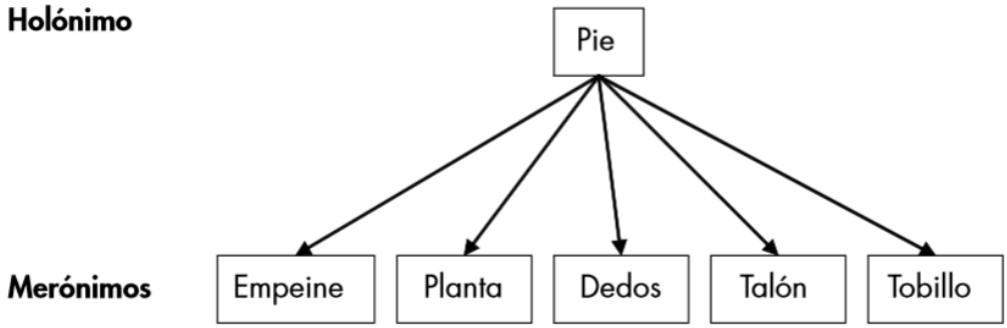


Figura 2.3 Holónimo Méronimos

2.3.3 Relación de exclusión y oposición

Las relaciones de exclusión y oposición se suelen conocer con el término común de antonimia, pero la oposición entre significados es diferente. En las relaciones de inclusión anteriormente descritas se mostró el ejemplo de álamo y árbol, donde los co-hipónimos (álamo, roble, pino, olivo, sauce …) tienen una parte única en su significado, lo que permite diferenciarlos unos de otros, pues bien, estas propiedades específicas se convierten en rasgos diferenciadores y mutuamente excluyentes, es decir, cada término tiene unas notas de significado tal que permite diferenciar sus rasgos de otro, por ejemplo un árbol no puede ser un álamo y un roble al mismo tiempo porque son completamente diferentes. (Escandell 2007)
Entonces se habla de oposición cuando la relación cumple ciertos requisitos, los cuales son descritos por Escandell (2007).

- Carácter binario

  Se considera oposición cuando solo los términos excluyentes son solo dos, por ejemplo, “dentro/fuera”, “abierto/cerrado”, son relaciones binarias.

- Carácter inherente

  Los términos se oponen de manera inherente y no accidental, por ejemplo, el movimiento por una línea vertical solo admite las direcciones de “arriba” y “abajo”, de modo que la oposición entre ambos términos es inherente.

- Carácter patente

  La oposición tiene que estar codificada de manera expresa, no implícita, por ejemplo, “ayer” y “mañana” son opuestos tomando el punto de referencia “hoy”.

Escandell (2007) identifica cuatro categorías de las relaciones de oposición, distribuidas de la siguiente manera, ver Figura 2.4.

Se denominan opuestos complementarios a los términos cuyos significados son excluyentes, entre los que no cabe un término medio, por mencionar algunos como verdadero/falso, vivo/muerto, par/impar, etc.

Los antónimos son opuestos que representan los extremos de una escala graduable, por lo que la oposición entre ellos es binaria inherente y patente, pero estos admiten términos medios, así como son de diferentes clases.

- Antónimos polares o monoescalares indican propiedades que pueden medirse de manera objetiva, por ejemplo, grande/pequeño, fuerte/débil, alto/bajo, corto/largo, además de caracterizarse por el hecho de que los términos que representan los extremos son graduales, estos admiten modificadores como poco, mucho, bastante, etc.
- Antónimos equipolentes o biescalares que atribuyen siempre propiedades en grado positivo, sensaciones perceptivas y emociones, entre estos se encuentran frio/caliente, dulce/salado, triste/alegre.
- Los inversos u opuestos expresan una misma relación desde perspectivas opuestas, por ejemplo, encima/debajo, comprar/vender, preceder/seguir.
- Los reversos son opuestos direccionales, tipos de oposiciones binarias basadas en un movimiento a partir de un punto dado, estos indican un movimiento que conduce a un cambio de estado de direcciones opuestas, como entrar/salir, abrir/cerrar.

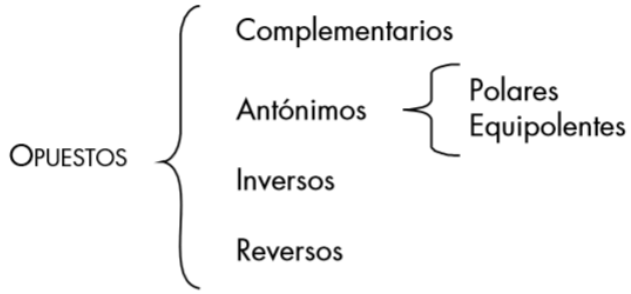


Figura 2.4 Relaciones de oposición

Como anteriormente lo mencionaron Karin (2013) y Cortez et al. (2014) los rasgos vistos son característicos de la semántica léxica, sin embargo, para la resolución de un problema está dentro del área de PLN.

La similitud textual entre pares de sentencias tiene muchas aplicaciones dentro del área de procesamiento de lenguaje natural, siendo asi los modelos de similitud semántica una de las mejores técnicas para la resolución de esta tarea, a continuación, se mencionan algunos.

2.4 Modelos de similitud semántica

Álvarez (2014) considera que los modelos formales de similitud son el modelo geométrico y el de teoría de conjuntos.

2.4.1 Modelo geométrico

El modelo geométrico, representa objetos en un espacio métrico $< X, \delta >$ logrando manipular el espacio geométrico de manera algebraica, donde cada objeto es representado por un punto $x \in X$ y δ (a, b) es la distancia entre un objeto a y un objeto b. Estas tareas se basan en función de distancia métrica que representa los objetos a, b, $c \in X$ como puntos en un espacio de coordenadas, a los cuales se asigna un número no negativo llamado distancia, entre menor sea el valor de la distancia, mayor es el grado de similitud. Como anteriormente lo define Tversky

(1977) en los axiomas de minimidad (Ecuación 1), simetría (Ecuación 2) y desigualdad del triángulo (Ecuación 3), que se describen en el tema 1.2 formulación del problema, pues los resultados son cuestionables.

2.4.2 Modelo de teoría de conjuntos

Tversky (1977) propone utilizar teoría de conjuntos para calcular la similitud de dos objetos, basándose en que la similitud debe describirse como una comparación de características, más que como una comparación de distancias métricas.

Para el dominio de objetos Δ = {a, b, c, ..., n}, donde cada objeto en delta está asociado con un conjunto de características especifico. Las características pueden corresponder a componentes tales como alguna parte del cuerpo, color o peso. La similitud se puede evaluar contando las características que tienen en común dichos objetos.

Sea S (a, b) una medida de la similitud de *a* hacia *b* para todas las distintas *a*, *b* en Δ. La escala *S* se trata como una medida ordinal de similitud. Es decir, *S (a, b) > S (c, d)* significa que *a* es más similar a *b* que *c* hacia *d*. La similitud de *a* hacia *b* se expresa como una función *F* de tres argumentos: $A \cap B$, las características que son comunes *a* y *b*; *A - B*, las características que pertenecen a pero no *a b*; *B - A*, las características que pertenecen a b pero no a (Tversky, 1977).

$$S(a,b) = F(A \cap B, A - B, B - A) \tag{4}$$

Donde $A \cap B$ son características de *a* y *b* que comparten en común; $A - B$ son características que tiene el objeto *A* pero no el objeto B; $B - A$ son las características que están en *B* pero no se encuentran en *A*. De esta forma se toman en cuenta las características que comparten dos objetos asi como los elementos que hacen diferentes a *A* de *B*, como a *B* de *A*. (Tversky, 1977; Álvarez, 2014)

2.5 Medición de similitud semántica

Los métodos que utilizan similitud semántica se pueden dividir en dos grupos: métodos basados en corpus (Velasco, 2013) los cuales utilizan grandes colecciones de datos para obtener estadísticas y calcular el grado de similitud entre las entradas proporcionadas, asi como el grupo de métodos basados en conocimiento (Baneaet al., 2013) los cuales utilizan conocimiento humano de un conjunto de datos, como es el caso de diccionarios donde se definen el significado de las palabras, por otro lado también existen los métodos híbridos que combinan los mencionados anteriormente (Álvarez, 2014, p.13).

2.5.1 Medidas de similitud basadas en caracteres

Álvarez (2014) describe que estas medidas de similitud miden la proximidad de los caracteres de cada palabra y pueden dividirse en los siguientes grupos:

- Subsecuencia común más larga. Algoritmo que calcula la similitud de dos cadenas, basado en la longitud de la cadena de caracteres más larga que exista
- Algoritmo de Levenshtein. Número mínimo de operaciones requeridas (insertar, eliminar, sustituir) para transformar una palabra en otra, el valor del resultado entre más alto, menos similar.
- Jaro-Winkler. Utiliza el número de caracteres que comparten las palabras, en la misma posición y transpuestas.

2.5.2 Similitud entre palabras considerando información semántica.

Estas medidas calculan la similitud semántica entre dos palabras a partir de una base de datos o un gran volumen de información.

- Basado en corpus

  Hiperespacio análogo al lenguaje. Crea un espacio semántico a través de coocurrencias de palabras, es decir, construir una matriz de palabra

por palabra que represente la palabra *i* con la palabra *j* en una colección de datos, el valor en cada entrada es acumulativo lo que significa que un valor alto para un par de palabras es referencia al mismo concepto.

Análisis semántico latente. Esta técnica asume que las palabras que semánticamente son similares coocurrirán en algunos fragmentos del texto, la información construye una matriz donde los renglones representa los párrafos del texto y las columnas las palabras, esto se compara de forma vectorial utilizando el coseno del ángulo que forman los vectores.

Información mutua. Utiliza métodos querys para calcular probabilidades de coocurrencia entre palabras.

- Basado en conocimiento, utiliza información derivada de redes semánticas. (Álvarez, 2014; Warin, 2004)

Resnik (res)

Devuelve un valor que denota que tan similares son dos palabras, según el contenido de información (IC) del subdominio menos común (nodo ancestral más específico), indicando que entre mayor sea el valor devuelto mayor es la similitud entre las palabras. Este cálculo está basado en el contenido de información en la red, es decir, a partir del nodo en común más cercano a las palabras que se están comparando. El resultado depende del corpus utilizado para generar el contenido de información y los detalles de cómo se creó el contenido de información, se define en **¡Error! No se encuentra el origen de la referencia.**.

$$sim = IC(lcs) \tag{5}$$

Donde $IC(lcs)$ es el nodo donde se encuentra el ancestro más cercano en común.

Lin (lin)

Retorna un valor continúo denotando la similitud entre un par de palabras y el ancestro más cercano y que sea común entre ellas. El algoritmo es útil en cualquier

dominio, siempre que pueda haber un modelo probabilístico para el dominio, a diferencia de Resnik, Jiang-Conrath y Leacock-Chodorow, quienes presuponen una taxonomía. La relación de este método está definida en la**¡Error! No se encuentra el origen de la referencia.**.

$$sim = \frac{2 * IC(lcs)}{IC(s1) + IC(s2)} \quad (6)$$

Donde $IC(s1)$ es el nodo en la red donde se encuentra la palabra uno, $IC(s2)$ es el nodo en la red donde se encuentra la palabra dos, $IC(lcs)$ es el nodo donde se encuentra el ancestro más cercano en común. Devuelve un valor de similitud en el rango de 0 a 1, donde 1 significa que los objetos son muy similares.

Jiang y Conrath (jnc)

Retorna un valor continúo denotando la similitud entre un par de palabras. Este método está basado en las palabras que se van a comparar y el ancestro más cercano a las dos palabras, esta puede ponderarse de tal forma que considere los factores de profundidad de los conceptos, ya que algunas áreas de una taxonomía son más densas (sus nodos tienen más hijos) que otras. El cálculo de esta función está definido en la **¡Error! No se encuentra el origen de la referencia.**.

$$sim = \frac{1}{IC(s1) + IC(s2) - 2 * IC(lcs)} \quad (7)$$

Donde $\mathrm{IC}(s1)$ es el nodo en la red donde se encuentra la palabra uno, $\mathrm{IC}(s2)$ es el nodo en la red donde se encuentra la palabra dos, $\mathrm{IC}(\mathrm{lcs})$ es el nodo donde se encuentra el ancestro más cercano en común. Devuelve un valor flotante que denota la similitud de los dos objetos Synset, un valor alto indica menos similitud entre dos objetos. Synset es un tipo especial de interfaz simple con agrupaciones de palabras sinónimas que expresan el mismo concepto.

Leacock y Chodorow (lch)

De este método se obtiene un valor que denota el grado de similitud entre dos palabras, basada en el cálculo de la ruta más corta entre las palabras en la red, pero, tomando en cuenta la profundidad máxima de la taxonomía de las palabras. Para esta medida se toma en cuenta que tan lejos se encuentra el ancestro común más cercano de las palabras comparadas. La función está definida en**¡Error! No se encuentra el origen de la referencia.**.

$$sim = -\log\left(\frac{p}{2d}\right) \tag{8}$$

Donde *p* es el camino más corto entre las palabras y *d* es la profundidad máxima de la taxonomía de las palabras en la red. El valor devuelto máximo es 3.583 lo que significa que es similar, esto varía según la profundidad de la taxonomía de los synset.

Wu y Palme (wup)

Muy similar a la medida anterior, también se calcula la ruta más corta entre las palabras comparadas. La diferencia recae en el uso de uno de los ancestros de la palabra en la red. En la medida anterior se utiliza el ancestro en común más cercano; en esta medida se utilizará el ancestro más cercano a alguno de las dos palabras. Esta medida se define en la **¡Error! No se encuentra el origen de la referencia.**.

$$sim = \frac{2 * Profundidad(LCS)}{Profundidad\big(IC(s1)\big) + Profundidad(IC(s2))} \tag{9}$$

Devuelve un valor flotante que denota la similitud de los dos objetos Synset, normalmente mayor que cero. Si no se puede encontrar una ruta de conexión entre los dos sentidos, se devuelve ninguno.

Path length (path)

Devuelve un valor continuo que denota que tan similares son dos palabras, con base en la trayectoria más corta que conecta las palabras en el árbol obtenido únicamente a través de relaciones del tipo hiponimia/hiperonimia. Esta medida se define en la **¡Error! No se encuentra el origen de la referencia.**.

$$sim = SP(IC(s1), IC(s2)) \quad (10)$$

Donde *SP()* es la función que devuelve el número de aristas del camino más corto de dos palabras en el árbol de WordNet. El resultado está en el rango de cero a uno, donde cero significa nula similitud y uno significa máxima similitud.

2.6 Medidas de similitud basadas en términos

Como se mencionó en el tema medidas de similitud basadas en términos un *n-grama* es la secuencia de *n* elementos, por lo tanto, la representación en un documento utilizando *n-gramas* va a dividir cada sentencia del documento en gramas de *n-términos.*

Modelo de unigrama

Aproxima la probabilidad de una palabra con la probabilidad de ocurrencia de esa palabra en el corpus de datos y sin tener en cuenta las palabras que la proceden.

Modelo de bigrama o digrama

Proporciona la probabilidad condicional de una palabra dada la palabra precedente.

Modelo de Trigrama

Tiene en cuenta las dos palabras anteriores.

Modelo de N-grama

Un modelo de n-grama es un modelo probabilístico que permite hacer una predicción estadística del próximo elemento de cierta secuencia, basándose en los *n* palabras anteriores (historia). En la ecuación 11 w_1^{i-1} representa la historia y w_i la predicción del modelo.

$$P\left(w_i \middle| w_1^{i-1}\right) = P\left(w_i \middle| w_{i-N+1}^{i-1}\right) \quad (11)$$

Las ventajas del modelo de N-grama son su simplicidad y contexto fácil de ampliar.

Distancia euclidiana

Se define como la distancia más corta entre dos puntos. Al utilizar esta distancia, los textos se deben representar en un espacio euclidiano mediante vectores, donde cada dimensión de los vectores representa una palabra de los textos, definida en **¡Error! No se encuentra el origen de la referencia.**.

$$dis_e(Vt_1, Vt_2) = \sqrt{\sum_{i=1}^{n} (Vt_{1i} - Vt_{2i})^2} \quad (12)$$

Donde:

- *N* es la cardinalidad del vocabulario existente para comparar ambas cadenas
- $Vt_1\ y\ Vt_2$ son los vectores que representan a las cadenas $t_1\ y\ t_2$
- $Vt_{1i}\ y\ Vt_{2i}$ son los i-ésimos elementos en los vectores que representan a las cadenas $t_1\ y\ t_2$

Distancia Manhattan

Distancia entre dos puntos que es la suma de las diferencias (absolutas) de sus coordenadas, utilizando los mismos vectores de distancia euclidiana, definida en**¡Error! No se encuentra el origen de la referencia.**.

$$dis_e(t1, t2) = \sum_{i=1}^{n} |Vt_{1i} - Vt_{2i}| \quad (13)$$

Similitud coseno

El resultado del coseno del ángulo entre dos vectores tendrá mayor similitud mientras menor sea el ángulo. El valor del ángulo resultante de entre dos vectores se define como en **¡Error! No se encuentra el origen de la referencia.**, con un valor de intervalo cerrado de -1 a 1.

$$\cos\theta = \frac{A \cdot B}{||A||\,||B||} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\,\sqrt{\sum_{i=1}^{n} B_i^2}} \tag{14}$$

Donde:

A, B son vectores de tamaño *N*

$A \cdot B$ producto punto de A por B

$||A||$ producto punto de A

$||B||$ producto punto de B

2.7 Similitud basada en traslape de palabras

Coeficiente de Dice

Basado en la teoría de conjuntos, toma el número de palabras que comparten un par de cadenas de caracteres y los divide entre el número total de la suma de ambas cadenas; el resultado es entre cero y uno, donde uno es máxima similitud, ver**¡Error! No se encuentra el origen de la referencia.**.

$$sim_D(t_1, t_2) = 2\frac{|t_1 \cap t_2|}{|t_1| + |t_2|} \tag{15}$$

Coeficiente de Jaccard

Se obtiene al dividir la intersección de términos entre la unión de los mismos, ver**¡Error! No se encuentra el origen de la referencia.**:

$$sim_J(t_1, t_2) = \frac{|t_1 \cap t_2|}{|t_1 \cup t_2|} \tag{16}$$

Coeficiente de traslape

Considera la cardinalidad de los caracteres que conforman el texto más pequeño, ver **¡Error! No se encuentra el origen de la referencia.**.

$$sim_T(t_1, t_2) = \frac{|t_1 \cap t_2|}{\min(|t_1|, |t_2|)} \quad (17)$$

Coeficiente de coseno

Divide la cardinalidad de la unión de dos conjuntos entre la raíz cuadrada del producto de las cardinalidades de los conjuntos considerados, ver **¡Error! No se encuentra el origen de la referencia.**.

$$sim_C(t_1, t_2) = \frac{|t_1 \cap t_2|}{\sqrt{|t_1||t_2|}} \quad (18)$$

2.8 Similitud semántica basada en recursos externos

Métodos estadísticos que estiman la probabilidad de una clase en vez de una palabra, los métodos clasificados de este apartado, se basan en la noción de que las palabras son próximas unas de otras en un diccionario u otro recurso lingüístico con el que trabaje su conjunto de datos.

2.9Técnicas de suavizado

Estos modelos tienen problemas de dispersión, es decir que existen una gran cantidad de eventos que son ignorados durante el proceso de entrenamiento y obtienen una frecuencia igual a cero (frases con alta probabilidad que contenga dicho evento), para evitar esto es necesario aplicar a los modelos las técnicas de suavizado smoothing, linear interpolation y backoff

- Smoothing

  Técnica para ajustar el modelo y estimar la probabilidad de mejor forma. Esta técnica tiende a unificar la distribución ajustando bajas/nulas probabilidades, asignando pequeños valores a este tipo de eventos.

- Linear interpolation

Esta técnica consiste en que las probabilidades se combinan con probabilidades de órdenes menores para intentar mantener la información de que palabras aparecen con una mayor frecuencia en el texto. Ejemplo: un modelo interpolado que combine probabilidades de unigrama (1-grama) y bigrama (2-grama)

- Backoff

Esta técnica también combina modelos de orden mayor con modelos de orden menor, utilizando según corresponda según el corpus de entrenamiento (generalmente utilizado en modelos de N-grama muy grandes).

2.10 Clasificador Naive Bayes

Es uno de los métodos de clasificación más utilizados, porque ofrece un análisis cualitativo de los atributos y valores que intervienen en el problema (importancia de los atributos). El aprendizaje basado en redes bayesianas es adecuado para las tareas de clasificación de textos mostrando mejores resultados que otros métodos (Malagón Luque Constantino, 2003).

Características:

- Cada ejemplo modifica la probabilidad de que la hipótesis sea correcta, lo que hace que la probabilidad estimada aumente o disminuya.
- Robusto frente al ruido de los conjuntos de entrenamientos que contienen datos incompletos o erróneos.
- Permite tener una predicción, esta depende de que los datos proporcionados sean suficientes para mostrar buenos resultados

Teorema de Bayes

$$P(C_i|D) = \frac{P(C_i)P(D|C_i)}{P(D)} \tag{19}$$

Donde:

- *D* es un documento del conjunto de datos de entrenamiento
- C_i es cada una de las posibles clases
- $P(C_i|D)$ es la probabilidad de que *D* pertenezca a la clase C_i

El clasificador Naive Bayes selecciona la clase que tenga la mayor probabilidad, pero este número es grande por lo que es necesario simplificar la expresión, utilizando la hipótesis de independencia condicional con el objetivo de poder factorizar la probabilidad.

"Los valores que describen un atributo de un ejemplo cualquiera *x* son independientes entre sí conocido el valor de la categoría a la que pertenecen. Así la probabilidad de observar la conjunción de atributos dada una categoría a la que pertenecen es justamente el producto de las probabilidades de cada valor por separado" (Malagón, 2003)

$$C_{NB} = \arg max\, P(C_i) \prod_{k=1}^{n} P(f_k|C_i) \tag{20}$$

Donde:

- f_k son las características del documento
- $P(f_k|C_i)$ es la probabilidad de ocurrencia de las características en la clase

Las diferentes implementaciones del algoritmo de Naive Bayes difieren en las aproximaciones de $P(f_k|C_i)$ y las técnicas de smoothing para el tratamiento de las probabilidades (Dubiau, 2013).

2.11 Support Vector Machines (SVM)

SVM es una clasificación binaria, un método supervisado de clasificación binaria basado en el entrenamiento de encontrar un hiperplano que separa las dos clases de puntos por un hiperplano, donde hay un vector de entrada y un vector de peso adaptativo, los vectores de palabras representan los conjuntos de datos. Dacheng Tao, Xiaoou Tang, Xuelong Li y Xindong Wu. (2006)

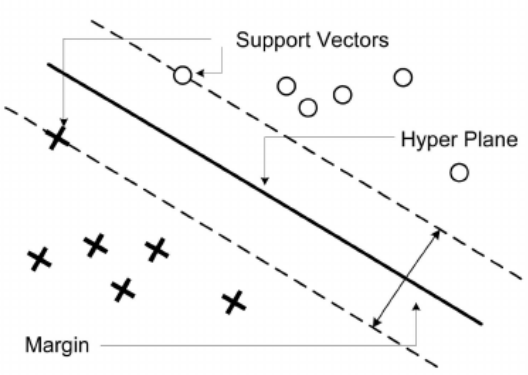


Figura 2.5 Support Vector Machine

En el espacio de características que representan el hiperplano optimo según la ecuación

$$\vec{w} \cdot \vec{x} + b = 0 \tag{21}$$

Donde:

- $\vec{w}$ es el vector normal de pesos perpendicular al hiperplano
- $\vec{x}$ es el vector de características
- b es el término independiente que nos permite elegir entre todos los vectores perpendiculares al vector normal

Al considerar un conjunto de datos de entrenamiento $D = \{(\vec{x}_i, \vec{y}_i)\}$ donde $\vec{x}_i$ es un vector de características del documento, $\vec{y}_i$ la clase y los valores resultantes serán +1 que indica que el documento pertenece a una clase y un valor -1 que pertenece a otra, lo que representa de qué lado del hiperplano se encuentra el vector que se quiere clasificar. (Dubiau, 2013).

$$f(\vec{x}) = sign\left(\sum_i \propto_i \overrightarrow{x_i} \cdot \vec{x} + b\right) \tag{22}$$

Dacheng et al. (2006) concluye que la retroalimentación de relevancia basada en las máquinas de vectores de soporte (SVM) es ampliamente utilizada, sin embargo, cuando el número de muestras de retroalimentación positiva es pequeño, el rendimiento del SVM se vuelve deficiente.

- el clasificador es inestable para un conjunto de entrenamiento pequeño
- el hiperplano óptimo de SVM puede estar sesgado

SVM encuentra un problema de sobreajuste

Capítulo 3 Estado del arte

En el siguiente capítulo se mostrará una revisión bibliográfica sobre el estado actual de las investigaciones de las herramientas, tecnologías y conjuntos de datos utilizados por los autores de sus respectivos trabajos, se empieza definiendo los métodos que se utilizan para la similitud entre pares de frases y palabras, pues algunos de estos métodos como se ha mencionado anteriormente en ocasiones no son eficientes y esto depende de su forma de clasificación o conjunto de datos que utiliza, es por esa razón que exponen a continuación sus trabajos relacionados con similitud semántica. También Krebs y Paperno (2016) indican que esta tarea tiene relación con la tarea de reconocimiento de imágenes para definir sus características, pues por medio de algunos métodos de comparación se pueden clasificar las imágenes con ciertos rasgos visibles que posee el objeto.

La composición semántica es la propiedad del lenguaje natural donde el significado de una expresión compleja es una función del significado de sus partes que la constituyen y del modo de su combinación. (Baroni, Bernardi y Zamparelli, 2014b)

A continuación, se describen los diferentes métodos que utilizaron los autores en sus trabajos de similitud y semántica léxica, así como sus comentarios de la implementación de dichos métodos en sus trabajos desarrollados.

3.1 Distribución semántica

Baroni et al. (2014b) menciona en su trabajo que la hipótesis distributiva establece que las palabras que aparecen en contextos similares (en el ámbito lingüístico) son semánticamente similares. Esto tiene un gran impacto en el área de computación, porque sugiere una forma práctica de recopilar automáticamente los significados de las palabras en gran escala, es decir si se puede equipar el significado con el contexto, podemos registrar los contextos en los que aparece una palabra en una colección de textos (un corpus) para crear un resumen de la historia distributiva de la palabra que luego se puede utilizar como un sustituto de su representación semántica. Los modelos semánticos distributivos, son los más directos en la

hipótesis distributiva en lineamientos computacionales. En un modelo semántico distributivo, cada palabra está representada por un vector matemático, es decir, que la palabra es vista como una lista ordenada de números, los valores en los componentes del vector son una función de la cantidad de veces que las palabras coocurren en la proximidad de varios contextos lingüísticos en un corpus.

Por ejemplo, supongamos que nuestro vocabulario contiene los sustantivos de perro (dog), hiena (hyena) y gato (cat), nuestros contextos son ladrar (barks) y correr (runs). A través del corpus descubrimos que perro tiene una proximidad a correr de 1 vez y más cercano a ladrar con 5 veces. Por lo tanto, podemos representar perro con el vector distributivo que constituye la primera columna de la Tabla 3.1. De manera similar, las hienas y gatos están representados por las siguientes columnas en la tabla 3.1, reflejando cuántas veces coocurren con las barras de respuesta en el corpus.

	perro	**hiena**	**Gato**
Correr	1	1	4
Ladrar	5	2	0

Tabla 3.1 Ejemplo de vector de palabras

Con base en esta evidencia se puede deducir que perro es más similar a hiena que a gato, porque ambos coocurren una vez con correr y varias veces con ladrar, mientras que gato se produce con mayor frecuencia con correr y ninguna vez con ladrar. Los vectores de distribución permiten una cuantificación lo más precisa posible de la similitud derivada de su representación como objetos geométricos. En la Figura 3.1, los vectores de las palabras se representan como segmentos orientados que se extienden desde el origen de un plano cartesiano a las coordenadas xy que corresponden a los valores en su primer y segundo componentes. En particular, los modelos semánticos distributivos suelen utilizar el coseno del ángulo formado por dos vectores como medida de similitud semántica.

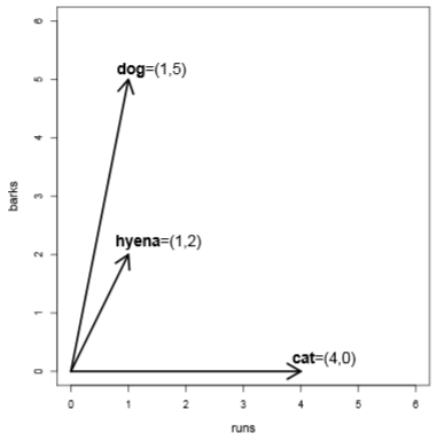


Figura 3.1 Representación gráfica de vectores de palabras

Baroni et al. (2014b) concluye que el utilizar distribución semántica trae una gran ventaja en la forma de manejar polisemia, pues no se necesitaran diferentes entradas para el mismo elemento, así como tener en cuenta la homonimia[1], pues este problema crea una diferencia de resultados muy notoria.

El coseno es una función del ancho del ángulo, y varía de 1 para vectores paralelos a 0 para vecinos perpendiculares (u ortogonales). Para el ejemplo anterior, perro tiene un coseno de 0.96 con hiena y un coseno más bajo de 0.2 con gato (Baroni et al. 2014b). La similitud de coseno generará una métrica que dice qué tan relacionadas están dos palabras o documentos al mirar el ángulo en lugar de la magnitud, como se puede observar en la Figura 3.2, si el ángulo de los vectores es muy cercano a 0 el coseno del ángulo será muy cercano a 1 por lo que las palabras son muy similares, para el caso de que el ángulo sea ortogonal (cerca de 90 grados) el coseno del ángulo será cercano a 0, lo que indica que las palabras no tienen relación alguna y para el caso de que los vectores sean opuestos (ángulo cercano a 180 grados) indica que el significado de las palabras es opuesto.

[1] Coincidencia en la escritura o en la pronunciación de dos palabras que tienen distinto significado y distinta etimología

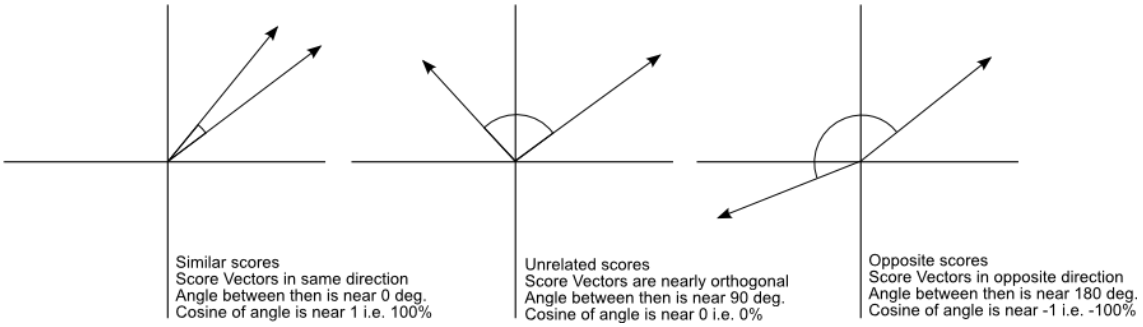


Figura 3.2 Valores de similitud de coseno

Los modelos semánticos distributivos codifican muchos más contextos dentro de cada palabra y como consecuencia funcionan con diferentes vectores de tamaño. Los matemáticos se refieren al conjunto de todos los vectores posibles de tamaño N como el espacio vectorial n-dimensional. Bajo esta definición, hay muchos espacios vectoriales distintos (posiblemente infinitos) de dimensiones N. (Baroni et al. 2014b)

3.1.1 Parámetros de modelos semánticos distributivos

La mayoría de las investigaciones sobre modelos semánticos distributivos se centran en los parámetros de los vectores distributivos extraídos desde un corpus, sin embargo, hay poca investigación sobre esto lo que afecta a los vectores resultantes, así lo mencionan "más datos son mejores datos" (Baroni et al. 2014b, p.250). Por lo que los diferentes contextos capturan diferentes tipos de similitud semántica o relación, ya que en cualquiera de los dos extremos puede capturar relaciones según el contexto, basado en palabras de co-ocurrencia o bien por relación semántica capturando relaciones taxonómicas, pero llegar al mejor modelo de un contexto aún es una tarea muy difícil.

Baroni et al. (2014b) concluyen que los modelos semánticos distributivos y sus componentes de un espacio semántico pueden interpretarse como una distribución en contextos en los cuales las palabras tienden a aparecer, sin embargo, a menudo la relación entre las palabras y contextos es indirecta. También menciona que los modelos semánticos distributivos son muy efectivos en simulación

psicológica, fenómenos lingüísticos relacionados a significados de palabras, predicción de similitud con conjuntos de datos hechos por humanos, categorizar conceptos nominales en hiperónimos, generar propiedades sobresalientes de conceptos y calidad de sustantivos, capturar intuiciones sobre la temática de los verbos o alteraciones del mismo.

3.2 Modelos de lenguaje semántico vectorial

Los modelos de lenguaje semántico vectorial representan cada palabra con un vector de valor numérico, estos vectores se pueden usar como características en una variedad de aplicaciones, tales como recuperación de información, clasificación de documentos, respuesta a preguntas, reconocimiento de entidades nombradas y análisis. La mayoría de los métodos de vectores de palabras se basan en la distancia o el ángulo entre pares de vectores de palabras como método principal para evaluar la calidad intrínseca del conjunto de representaciones de palabras. (Pennington J., Socher R. & Manning C. D. 2014)

Las dos familias de modelos principales para aprender vectores de palabras son:

- Métodos de factorización de matriz global, como el análisis semántico latente (LSA)
- Métodos de ventana de contexto local, como el modelo de skip-gram de Mikolov

Actualmente, ambas familias sufren importantes inconvenientes. Si bien los métodos como LSA aprovechan eficientemente la información estadística, funcionan relativamente mal en las palabras de tarea de analogía, lo que indica una estructura de espacio vectorial subóptima. Métodos como skip-gram puede funcionar mejor en la tarea de analogía, pero utilizan pobremente las estadísticas del corpus ya que se entrenan en contextos locales separados en lugar de los recuentos globales de co-ocurrencia. (Pennington et al. 2014)

Se ha prestado mucha atención a la cuestión de si las representaciones de palabras distributivas se aprenden mejor a partir de métodos basados en conteos o en métodos basados en predicciones. Los modelos basados en predicción obtienen

un apoyo sustancial; por ejemplo, Baroni et al. (2014a) sostienen que estos modelos funcionan mejor en una variedad de tareas.

Los resultados de Berg-Kirkpatrick, T., Burkett, D., & Klein, D. (2012) revelan que la relación entre la ganancia métrica y la significación estadística es compleja, por lo tanto, los umbrales simples no reemplazan las pruebas de significación. Sugieren el uso de pruebas de significación estadística para validar las ganancias métricas de PLN, pero también que en la discusión popular surgen reglas de oro informales y para algunas configuraciones, cuando los sistemas anteriores no están disponibles, los resultados empíricos pueden complementar pruebas no emparejadas en la evaluación del progreso.

Pennington (2014) argumenta que en su trabajo las dos clases de métodos no son tan diferentes, ya que ambos sondean las estadísticas subyacentes de co-ocurrencia del corpus, pero la eficiencia de los métodos basados en el conteo global estadísticos pueden ser ventajoso, modelos como CBOW y word2vec. (Baroni Marco and Dinu Georgiana and Kruszewski Germán., 2014a)

3.3 Modelos de clasificación discriminativos y generativos

Los clasificadores generativos aprenden el modelo a partir de la forma en que se generan las observaciones, a partir de la probabilidad conjunta $P(x, y)$ y realiza la predicción maximizando la probabilidad condicional $P(y \mid x)$ que es calculada con el teorema de Naive Bayes. Los clasificadores discriminativos (condicionales) modelan directamente la probabilidad condicional $P(y \mid x)$ para realizar la predicción a partir de la estructura de los datos sin tener en cuenta la forma en que se generan. (Ng, A. & Jordan, M., 2001)

El modelo generativo tiene un error asintótico más alto que el modelo discriminativo, esto se debe a medida que aumenta el número de ejemplos de entrenamiento, pero el modelo generativo también puede aproximarse a su error asintótico mucho más rápido que el modelo discriminativo. La desventaja de los modelos generativos es que resuelven un problema más general como paso

intermedio, por otro lado, en los modelos discriminativos se resuelve el problema de clasificación directamente (Ng & Jordan, 2001).

Las predicciones teóricas de Ng & Jordan se confirman; en algunos casos Bayes no alcanza el rendimiento, pero esto se puede observar principalmente en conjuntos de datos particularmente pequeños en los que los ejemplos de entrenamiento probablemente no pueden crecer lo suficiente. Berg-Kirkpatrick, et al. (2012) menciona que al diseñar una tarea compartida es importante saber qué tan grande debe ser el conjunto de pruebas para que las pruebas de significación sean sensibles a pequeñas ganancias en la métrica de rendimiento.

Los modelos generativos más importantes son los modelos de N-Gramas y Naive Bayes. Los modelos discriminativos más relevantes son Logistic Regression, Maximum Entropy Models, SVM, etc. (Dubiau Luciana, 2013)

Muchos sistemas y técnicas actuales de PLN tratan las palabras como unidades atómicas, no existe una noción de similitud entre las palabras, ya que se representan como índices en un vocabulario. Esta elección tiene varias buenas razones: simplicidad, robustez y la observación de que los modelos simples entrenados en grandes cantidades de datos superan a los sistemas complejos entrenados con menos datos. Un ejemplo es el popular modelo de N-gramas utilizado para el modelado estadístico de lenguaje. (Mikolov, T., Chen, K., Corrado, G. & Dean, J., 2013a)

3.4 Vectores de palabras

Los modelos de PLN estadísticos así como los modelos basados en reglas se refieren a las palabras como símbolos atómicos, representando cada palabra como un vector $\mathbb{R}^{|V|}$ donde V es el vocabulario de la lengua, que posee 1 en el índice del vector de la palabra en el vocabulario ordenado y 0 en cualquier otro caso. La palabra se representa como una entidad completamente independiente por lo que en algunas ocasiones no se obtiene ninguna noción de similaridad, además de que el vocabulario de la lengua es de grandes dimensiones. Una palabra está representada por medio de palabras que la rodean, esto es conocido como matriz

de co-ocurrencia en PLN siendo a nivel de documento, es decir que representan cada documento como la cantidad de ocurrencias de determinada palabra en el documento que es para tópicos generales, o bien a nivel de ventana de palabras que muestran las ocurrencias de las palabras que tiene alrededor de una palabra central, estas son ideales para capturar información sintáctica y semántica.

Mikolov, T., Yih, W.t. & Zweig, G. (2013b) menciona que las representaciones de palabras del espacio vectorial se aprenden implícitamente por los pesos de la capa de entrada del modelo de lenguaje de red neuronal; estas representaciones son sorprendentemente buenas para capturar regularidades sintácticas y semánticas del lenguaje, cada relación se caracteriza por un desplazamiento vectorial específico de la relación, permitiendo un razonamiento orientado a vectores, basado en las compensaciones entre palabras. De igual manera describen que una característica de los modelos de lenguaje de red neuronal es su representación de palabras como vectores de valores numéricos reales de alta dimensión. En estos modelos, las palabras se convierten a través de una tabla de búsqueda aprendida en vectores de valor numérico real que se utilizan como entradas en una red neuronal. Una de las principales ventajas de estos modelos es que la representación distribuida alcanza un nivel de generalización que no es posible con los modelos de lenguaje clásico, un modelo espacial continuo funciona en términos de vectores de palabras donde es probable que palabras similares tengan caracteres similares. Por otro lado, Oded Avraham & Yoav Goldberg. (2017) mencionan que los modelos de inserción de palabras aprenden un espacio de representaciones de palabras continuas, en las que se espera que palabras similares estén cerca unas de otras, los términos semejantes se refieren a similitud prematura. Por lo tanto, cuando los parámetros del modelo se ajustan en respuesta a una palabra o secuencia de palabras en particular, las mejoras se trasladarán a ocurrencias de palabras y secuencias similares.

Mikolov et al. (2013a) menciona que es posible entrenar vectores de palabras de alta calidad utilizando arquitecturas de modelos muy simples, en comparación con los modelos de redes neuronales populares (tanto de avance como recurrentes). Debido a que la complejidad computacional es mucho más baja, es

posible calcular vectores de palabras de alta dimensión muy precisos a partir de un conjunto de datos mucho más grande.

Por otra parte, Vylomova, E., Rimell, L., Cohn, T., & Baldwin, T. (2016) mencionan que en trabajos de investigación se ha demostrado que la simple resta de vectores sobre incrustaciones de palabras es sorprendentemente efectiva para capturar diferentes relaciones léxicas, a pesar de no tener una supervisión explícita. Las incrustaciones de palabras, son vectores densos y de baja dimensión que se obtienen a partir de una red neuronal basada en predicciones, para contextos de palabras. Mediante el uso de la agrupación demostraron que la diferencia de vectores captura muchos tipos de diferencias morfosintácticas y morfosemánticas, pero que las relaciones semánticas léxicas se capturan menos, la clasificación sobre la diferencia de vectores funciona bien en un entorno no supervisado, en el que el clasificador solo encuentra pares de palabras que corresponden a algunas de las relaciones definidas, además el muestreo negativo mejora la clasificación cuando el entrenamiento y el vocabulario de la prueba se dividen para minimizar la memoria léxica.

Lai, S., Liu, K., Xu, L., & Zhao, J. (2016) mencionan en su trabajo que la mayoría de los métodos de entrenamiento para formar incrustaciones de palabras se basan en la misma hipótesis distributiva que dice que las palabras que aparecen en contextos similares tienden a tener significados similares, en consecuencia, los diferentes métodos modelan su relación entre una palabra objetivo (w) y su contexto (c) incrustadas en diferentes vectores, asi que los métodos existentes varían en dos aspectos principales:

- La relación entre la palabra y su contexto
- La representación del contexto

La habilidad de entrenar precisas incrustaciones de palabras es difícil, pues sus resultados están relacionados con el corpus de entrenamiento y la tarea a desarrollar.

Los siguientes son modelos de incrustaciones de palabras:

Neural Network Language Model (NNLM)

Yoshua et al. proponen este modelo que aprende simultáneamente una inserción de palabras y un modelo de lenguaje, para cada muestra en el corpus maximiza la probabilidad de la última palabra, este modelo usa la concatenación de la previa incrustación de palabras en la entrada.

$$x = [e(w_1), \dots, e(w_{n-2}), e(w_{n-1})] \tag{23}$$

La estructura del modelo es una red neuronal de avance con una capa oculta:

$$h = \tanh(d + Hx) \qquad y = b + Uh \tag{24}$$

Donde *U* es una matriz de transformación, b y d son vectores de polarización. El paso final es aplicar una capa de softmax para obtener la probabilidad de la palabra objetivo. Softmax es una función que proporciona probabilidades para cada clase posible en un modelo (como se cita en Lai et al. 2016).

Log-bilinear Language Model (LBL)

Este modelo es similar al NNLM. El modelo LBL utiliza una función de energía log-bilineal que es casi igual a la del NNLM y elimina la función de activación no lineal tanh

Collobert and Weston (C&W)

Este modelo solo entrena la palabra de incrustación y no hace predicción de una palabra, combina la palabra objetivo y su contexto para evaluarlos. La entrada es la concatenación de las palabras destino y de contexto; el entrenamiento maximiza la puntuación de la secuencia de un corpus mientras minimiza la puntuación de ruido.

$$max(0,1 - s(w, c) + s(w', c)) \tag{25}$$

La palabra objetivo *w* se remplaza con una palabra aleatoria w´ del vocabulario.

CBOW and Skip-gram

Estos modelos intentan minimizar la complejidad computacional. CBOW utiliza la inserción promedio de las palabras de contexto como representación del contexto,

y skip-gram usa una de las palabras del contexto como la representación del contexto; ambos modelos necesitan la información del orden de las palabras y solo usan regresión logística para predecir la palabra objetivo.

Los métodos anteriores mostraron las formas de solución que utilizaron algunos autores para la tarea de similitud entre pares de palabras; a continuación, se muestran trabajos relacionados con el tema de atributos discriminativos, retomando a Tversky (1977) con su estudio de “Features of similarity” como anteriormente se ha mencionado, dice que “la similitud debe describirse como una comparación de características, más que como una comparación de distancias métricas” pues es uno de los trabajos en los que varios autores tomaron referencia del tema de atributos discriminativos.

Al día de hoy los trabajos relacionados a características de un objeto (atributos discriminativos) son principalmente de reconocimiento de imágenes, como lo describen Wang, Y., & Mori, G. (2010) este tipo de modelos capturan la relación entre atributos de un objeto y el objeto mismo, es por esa razón que su modelo relacionado se basa en clasificación de un objeto a partir de etiquetas, las cuales representan los objetos por sus categorías o nombres (pájaro, manzana, silla, etc.) y a su vez estos mismo se describen a partir de propiedades o atributos (tiene pico, rojo, hecho de madera, etc.), estos se dividen en categorías de partes (tiene oído), forma (redondo), material (metal), color (rojo), etc. Los autores mencionan que a este tipo de etiquetado se le llama attribute-centric lo cual ayuda a mejorar el modelo, para aplicaciones prácticas pues se puede reconocer en ambos sentidos de objeto – atributo y atributo – objeto; su modelo descrito utiliza lo siguiente:

- Conjunto de datos
    - a-Pascal[2] que contiene 6340 imágenes con 20 objetos y sus respectivas etiquetas
    - Yahoo[3] con 12 objetos y 64 atributos.

[2] http://host.robots.ox.ac.uk/pascal/VOC/
[3] https://webscope.sandbox.yahoo.com/catalog.php?datatype=i

- Grafo de relación de atributos para relacionar los atributos con el objeto a clasificar

Este trabajo encapsula la relación entre los diferentes atributos y su grafo de clasificación.

Duan, K., Parikh, D., Crandall, D.J., & Grauman, K. (2012) mencionan en su trabajo que los atributos son conceptos visuales que pueden ser detectados por máquinas, entendidos por humanos y compartidos a través de categorías, pues los atributos discriminativos ayudan a exponer los detalles que componen a un objeto haciendo fácil su comprensión, sin embargo, esta tarea se ha centrado en objetos específicos pues la clasificación de sus categorías con sus respectivos atributos ha incrementado de gran manera. De manera que la mayoría de los atributos semánticamente significativos corresponde con las partes semánticas de los objetos, viendo asi a un atributo como una propiedad para comprender parte de un objeto. Guo, Y., Ding, G., Jin, X., & Wang, J. (2015) en su trabajo dicen que los atributos pueden ser una conexión entre características visuales y definiciones semánticas de alto nivel, sin embargo, se tiene que utilizar una forma predecible y discriminativa simultáneamente, para lograr el reconocimiento de un objeto, lo que conlleva a tener un conjunto de datos muy grande y estricto.

Los autores Lazaridou, A., Pham, N. T., & Baroni, M. (2016) confirman con su trabajo resultados exitosos utilizando atributos discriminativos predictivos los cuales se basan en los objetos representados por imágenes naturales, donde el agente puede extraer sus características más relevantes desde datos reales. El conjunto de datos utilizado para el desarrollo de su trabajo consiste en pares de referencias y contextos con respecto al referente que debe estar identificado por sus respectivos atributos; este conjunto descrito cuenta con 500 conceptos y 636 atributos generales (no se toman en cuenta conceptos ambiguos).

La propuesta de los autores Lazaridou et al. (2016) es aprender a predecir los atributos discriminativos a partir de su objeto y contexto, es decir que se debe aprender cuando un atributo está activo para un objeto en específico, asi como es crucial determinar cuando el atributo es discriminativo de un par de objetos, esta función utiliza instancias visuales de vectores (ConvNet), neural network que toma

la entrada de una imagen para predecir su vector, para la supervisión a nivel de atributo por medio de diferencia simétrica del vector.

Los modelos anteriormente mencionados consisten en un conjunto de datos basado en imágenes que tiene una colección de objetos con sus respectivos atributos en forma de etiquetas las cuales ayudan a clasificar de mejor manera las características de un objeto, sin embargo como lo mencionaron los autores, estos conjuntos de datos están basados en un tema en específico, además de que para el entrenamiento es necesario contar con un gran número de imágenes para que el modelo entregue buenos resultados. Como lo mencionan Lazaridou et al. (2016) una propuesta para el tema anterior es aprender a predecir los atributos de objeto, así como supervisar sus características a nivel de similitud de vectores, pues esto ayudará a obtener mejores resultados durante las pruebas.

Krebs y Paperno (2016) toman en cuenta los trabajos de reconocimiento visual porque como demostraron Faruqui et al. (2016) los problemas a nivel semántico y computacional son variados en cualquier tipo de similitud de palabras y necesitan mejores soluciones, es así que tomando como referencia el trabajo y conclusiones de Lazaridou et al. (2016) proponen la tarea de captura de atributos discriminativos la cual menciona que los modelos de similitud de palabras, ya sea por formula de similitud, vectores de palabras, reconocimiento de imágenes e incluso semántica léxica mostrarán resultados variados, pues su conjunto de datos es muy variado, por lo que su propuesta de hacer uso de las características que definen a cada objeto (atributos discriminativos) ayudara a mejorar la similitud entre dos objetos de una manera comprensible; esta tarea es retomada en SemEval 2018.

3.5 Contribución

La contribución de esta tesis es la comparativa de los resultados obtenidos por los diferentes métodos existentes, demostrando que los problemas a nivel semántica y computacional son cercanos entre si, un ejemplo de ello como se menciono anteriormente son los problemas como la polisemia, pues estos son muy

difíciles para ambas ramas de la investigacion. Es por esa razón que al recopilar información de los métodos mencionados se encontró que en el área de la semántica lexica estudian al lenguaje dentro de dos categorías, las cuales son lenguaje formal y lenguaje natural, este ultimo se encuentra en constante cambio y no tiene alguna regla o norma que lo defina, por lo que es complejo estandarizar un método para definir a un lenguaje natural.

Muchos métodos de PLN han hecho pruebas al realizar esta tarea, intentando formalizar al lenguaje natural en un corpus o serie de reglas, sin embargo, un lenguaje siempre esta en constante cambio; es por eso que el desarrollo del modelo propuesto se desarrolla con los principios de semántica lexica, los cuales son definir una palabra con una colección de palabras que existen en un diccionario; incluyendo los principios de vectores de palabras para aproximar los resultados de una búsqueda especifica con elemenros similares, comunes y relacionados.

Entonces el modelo propuesto aporta la comparación entre los diferentes modelos y conjuntos de datos existentes de la tarea de similitud entre pares de palabras dentro del PLN, mostrando en los resultados de la sección de conclusión que los métodos de similitud por medio de una formula no obtienen buenos resultados en la identificación de captura de atributos discriminativos, ya que no pueden clasificar la mayoría de los elementos dentro de la tarea, los modelos a partir de vectores de palabras multidimensionales y de contexto mostraron una mejor precisión, sin embargo, los resultados dependen del conjunto de datos que se utiliza para cada uno.

Por esta razón el modelo propuesto maneja los principios de semántica lexica es decir, que la palabra es definida a partir de otra palabra la cual tiene un significado dentro de un diccionario y este puede definir a los objetos a partir de la definición de la palabra, utilizando definiciones de diccionarios, relaciones de semejanza (sinónimos), relaciones de inclusión (Hiponimia, Hiperonimia, Co-Hiponimia, Meronimia, Co-Meronimia), relaciones de exclusión y oposición (antonimos), brindando mejores resultados en la clasificación de muestras.

Capítulo 4 Diseño

En el siguiente capítulo se describe en que consiste la tarea de atributos discriminativos, su conjunto de datos utilizado, herramientas y librerías de Python utilizadas, diseño del desarrollo del modelo propuesto, descripción del algoritmo principal y explicación de algunos métodos implementados.

En contraste con la similitud, la naturaleza de la diferencia semántica entre dos palabras (relacionadas) puede variar mucho. En los modelos de similitud actuales la diferencia en su modelado puede ayudar a capturar aspectos individuales del significado de cada palabra, por lo tanto, esta tarea por si sola es demasiado compleja al evaluar sus representaciones semánticas por lo que no es suficiente su progreso para impulsar los modelos computacionales de hoy en día. Una posible solución a este problema es propuesta por Krebs y Papermo en 2016 la cual consiste en extraer diferencias semánticas entre un par de palabras, al referirse a sus atributos que componen a cada una, de manera que una diferencia se puede expresar como la presencia o ausencia de un atributo en específico, asi que ocupar atributos discriminativos puede brindar mejores resultados que los modelos de similitud existentes, además de tratar las palabras como objeto o entidad lo cual proporciona la ventaja de utilizar etiquetas de clasificación para diferenciar la información del conjunto de datos y poder predecir resultados más fiables. (Krebs y Paperno, 2016; Wang y Mori, 2010; Guo, Ding, Jin y Wang 2015; Lazaridou, The Pham y Baron, 2016)

4.1 Descripción de la tarea

La tarea de Captura de atributos discriminativos consiste principalmente en clasificar un par de palabras de un conjunto de datos, por lo que dadas dos palabras, se tiene que saber cuál es su diferencia entre significados, esta clasificación se calcula a partir de un trio de palabras las cuales son una palabra pivote, una palabra de comparación y un atributo discriminativo de alguna de las anteriores permitiendo

clasificar si entre una palabra y otra existe una característica que define a su significado, es decir un atributo discriminativo.

Entonces para cada par de palabras se verifica que la *palabra1 (pivote)* tenga relación con el atributo proporcionado y la *palabra2 (comparación)* no tenga relación alguna con el *atributo*, para agregarlo a la lista de ejemplos candidatos positivos, esta lista cuenta con un conjunto de datos de las palabras con sus respectivos atributos discriminativos. Algunos elementos son:

- palabra1 (pivote)
- palabra2 (comparación)
- atributo discriminativo
- etiqueta

  Valor de salida 0/1 que indica si el par de palabras es atributo discriminativo
- Lista ejemplos positivos

  Lista de palabras con ejemplos positivos de atributos discriminativos, palabra y característica tienen relación entre sí.
- Lista ejemplos negativos

  Lista de palabras con ejemplos negativos, palabra y característica no tienen relación alguna.

Un ejemplo de lo anterior seria *palabra1* airplane (avión), *palabra2* helicopter (helicóptero) y *atributo* wings (alas), donde la *palabra1* tiene relación con el atributo, pero la *palabra2* no tiene relación alguna, ya que un helicóptero no tiene alas, tiene hélices, entonces para el caso contrario de *palabra1* helicóptero, *palabra2* avión, *atributo* alas, se tomará en otra lista, una lista de ejemplos negativos, ya que la *palabra1* no tiene relación con el *atributo*.

palabra1	*palabra2*	*atributo*	etiqueta	Lista
airplane	helicopter	wings	1	positivos
helicopter	airplane	wings	0	negativos

Para la lista de ejemplos negativos se toman en cuenta los casos:

- ambas palabras (*palabra1 y palabra2)* tengan relación de similitud con el *atributo.*

palabra1	*palabra2*	*atributo*	etiqueta	Lista
banana	pineaple	yellow	0	negativos

- *palabra2* tiene relación de similitud con el *atributo*, pero no la *palabra1* (el conjunto de datos está construido de tal manera que el *atributo* de cada elemento siempre pertenece a *palabra1*).

palabra1	*palabra2*	*atributo*	etiqueta	Lista
banana	apple	red	0	negativos

- Ambas palabras no tienen relación alguna con el atributo, para este último caso la cantidad de ejemplos es muy grande.

palabra1	*palabra2*	*atributo*	etiqueta	Lista
banana	pineaple	red	0	negativos

Los conceptos que tienen diferentes significados han sido desambiguados, es decir que las palabras con diferentes significados no son tomadas en cuenta. Por ejemplo, para las palabras bow_ (arma) y bow_ (cinta).

4.2 Conjunto de datos

Dentro del conjunto de datos proporcionado por CodaLab se encuentra una estructura de palabras, con el siguiente orden *palabra1*, *palabra2*, *atributo* y *etiqueta*, donde etiqueta es la asignación binaria (0/1) que identifica si la línea de palabra-atributo es un atributo discriminativo o no, de igual manera el conjunto de datos de validación (validation) esta verificado por jueces humanos, por lo que los resultados de este conjunto tienen menos ruido que el conjunto de entrenamiento (train), ya que este último es generado automáticamente e incluye ejemplos

positivos y negativos (mayormente). La información de los conjuntos de datos de entrenamiento, validación y pruebas (test) se encuentran detalladas en la Tabla 4.1.

Cabe mencionar que el conjunto de pruebas no contiene etiquetas de clasificación, esto con el fin de ajustar los parámetros con los conjuntos de datos de entrenamiento y validación para asi poder evaluar la propuesta del modelo en la plataforma en línea de CodaLab, la cual calificará de forma automática los resultados obtenidos del modelo, mostrando el porcentaje total de aciertos obtenidos en la sección de resultados, esto con el fin de participar en la resolución de la tarea de atributos discriminativos.

Conjunto de datos	Ejemplos	Neg	Pos	Atributos	Neg	Pos
Entrenamiento	17,782	11191	6591	1,292	1290	333
Validación	2,722	1358	1364	576	383	410
Prueba	2,340			577		

Tabla 4.1 Cantidad de ejemplos de los conjuntos de datos

4.3 Ruido en datos

Durante la extracción de datos para el entrenamiento se encontró que había mucho ruido en los datos proporcionados por lo que no eran consistentes, es decir que para el conjunto de entrenamiento (training) se encuentran entradas que no pueden ser clasificadas correctamente, una de las razones es que su orden de magnitud es mayor, dado que la mayoría de los ejemplos son negativos, además de ser generado automáticamente por lo que algunas entradas pueden ser incorrectas, esto con el objetivo de tener un entrenamiento del sistema rico en parámetros, por lo que se tiene mayor cuidado al clasificar cada línea de los conjuntos de datos.

4.4 Fuentes de conocimientos externas

4.4.1 WordNet

WordNet[4] se creó y se mantiene en el Laboratorio de Ciencia Cognitiva de la Universidad de Princeton bajo la dirección del profesor George A. Miller. WordNet

[4] https://wordnet.princeton.edu/

es una extensa base de datos léxica para el idioma inglés, la cual agrupa las palabras en inglés en conjuntos de sinónimos llamados synsets, los cuales contienen sustantivos, verbos, adjetivos y adverbios, estos proporcionan definiciones cortas y generales, las cuales registran las diversas relaciones semánticas entre los conjuntos de sinónimos. Los synsets están interconectados por medio de relaciones conceptuales-semánticas y léxicas. (Miller et al., 1990)

Las relaciones semánticas que se establecen de foma general entre synsets son las siguientes:

- Sinónimos
- Antónimos
- Hiponimia/Hiperonimia
- Meronimia/Holonimia
- Adjetivos raíz
- Implicación y causa

La principal relación entre las palabras en WordNet es la sinonimia, los sinónimos, palabras que denotan el mismo concepto y son intercambiables en muchos contextos, se agrupan en conjuntos desordenados (synsets). Cada uno de los 116000 synsets de WordNet está vinculado a otros synsets por medio de un pequeño número de relaciones conceptuales. Las formas de palabras con varios significados distintos (polisemia) se representan en synsets distintos, por lo tanto, cada par de synset-significado en WordNet es único. (Miller, 1995)

4.4.2 NLTK

NLTK[5], el kit de herramientas de lenguaje natural, es un conjunto de módulos de Python que proporciona muchos tipos de datos de PLN, tareas de procesamiento, muestras de corpus y lectores, junto con algoritmos de ejemplo, tutoriales, asi como foros de discusión y conjuntos de problemas. Proporciona interfaces fáciles de usar para más de 50 recursos corporales y léxicos como WordNet, junto con un conjunto de bibliotecas de procesamiento de texto para clasificación, tokenización,

[5] https://www.nltk.org/

derivación, etiquetado, análisis y razonamiento semántico, envoltorios para bibliotecas de PNL de gran solidez. (Bird, 2006)

4.4.3 Spacy

SpaCy[6] es una biblioteca de software de código abierto para el procesamiento avanzado de lenguaje natural, escrita en los lenguajes de programación Python y Cython; está diseñado para uso de producción y ayuda a crear aplicaciones que procesan y comprenden grandes volúmenes de texto. Se puede utilizar para construir sistemas de extracción de información o de comprensión del lenguaje natural, o para preprocesar texto para un aprendizaje profundo.

Sus características principales son:

- Tokenización no destructiva
- Reconocimiento de entidad nombrada
- Soporte para más de 34 idiomas
- 13 modelos estadísticos para 8 idiomas
- Vectores de palabras pre-entrenados
- Fácil integración de aprendizaje profundo
- Etiquetado de parte del discurso
- Análisis de dependencias etiquetadas
- Segmentación de oraciones impulsada por la sintaxis
- Construido en visualizadores para sintaxis
- Conveniente asignación de cadena a hash
- Exportar a numerosos arreglos de datos
- Serialización binaria eficiente
- Fácil modelo de empaquetado y despliegue
- Velocidad del estado de la técnica
- Precisión robusta, rigurosamente evaluada

[6] https://spacy.io/

spaCy es un marco de NLP moderno y confiable en el estándar para hacer NLP con Python, sus principales ventajas son velocidad, precisión y extensibilidad, además de que contiene activos útiles como incrustaciones de palabras.

4.4.4 Sense2vec

Sense2vec[7] es parecido a word2vec con la diferencia de que sense2vec permite aprender vectores de palabras más interesantes, detallados y sensibles al contexto, es decir que dado un corpus etiquetado (ya sea a mano o por un modelo) con una o más etiquetas por palabra, el modelo de sense2vec primero cuenta el número de usos de cada palabra y genera una inclusión de sentido al azar para cada uso, luego, se entrena a un modelo utilizando las configuraciones de modelo CBOW, Skip-gram o Structured Skip-gram, ver Figura 4.1 (Trask, Michalak, & Liu, 2015).

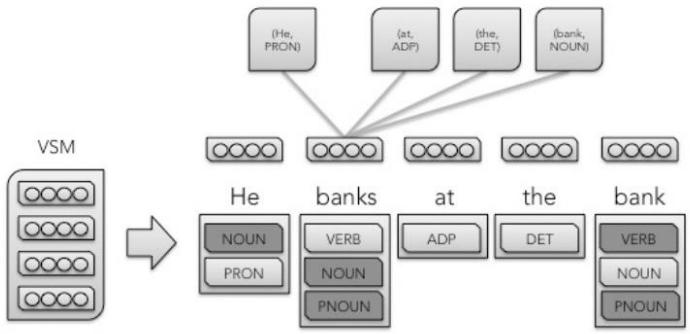


Figura 4.1 Representación gráfica de Sense2vec

Esta biblioteca es una implementación simple de Python / Cython para cargar y consultar modelos de sense2vec. Si bien se usa mejor en combinación con spaCy, la biblioteca de sense2vec es muy liviana y también puede usarse como un módulo independiente.

[7] https://github.com/explosion/sense2vec

4.4.5 Diccionarios

Con la ayuda de las librerías Beautiful Soup y request se puede acceder a los resultados de los siguientes diccionarios en línea:

Dictionary.com[8] es la principal fuente en línea del mundo de definiciones en inglés, sinónimos, orígenes de las palabras, pronunciaciones de audio, oraciones de ejemplo, frases de jerga, idiomas, juegos de palabras, términos legales y médicos, con sus servicios digitales gratuitos.

Como principal diccionario de sinónimos en la web, Thesaurus.com[9] ofrece a los usuarios más de 550,000 sinónimos y un conjunto de herramientas que simplifican el proceso de escritura. Los filtros de relevancia, complejidad y duración exclusivos, ayudan a encontrar rápidamente la palabra perfecta, además de la capacidad de filtrar los resultados de búsqueda por relevancia, longitud de palabra y complejidad.

4.4.6 Wikipedia-Api

El objetivo de Wikipedia-Api[10] es proporcionar una API simple y fácil de usar para recuperar información de la página de Wikipedia. Algunas de sus principales funciones son la extracción de textos, secciones, enlaces, categorías, traducciones, etc.

4.5 Preparación de los datos

Con el objetivo de mejorar los resultados del clasificador y utilizar el máximo posible de ejemplos de los conjuntos de datos de entrenamiento y validación se hicieron algunas transformaciones en el manejo de información del clasificador, algunas de estas son:

[8] https://www.dictionary.com/
[9] https://www.thesaurus.com/
[10] https://wikipedia-api.readthedocs.io

- Normalizar las palabras, todos los caracteres en minúscula, eliminación de caracteres especiales, separación de palabras compuestas. Ejemplo: Manzana_01 = manzana

- Lematizacion de las palabras unificar los términos para una palabra a partir de su lemma eliminando conjugaciones, grado, persona, género, número, etc. Ejemplo: lemma(empieza)=empezar

- Tokenización separación de sentencias y eliminación de stopwords en este caso para las herramientas de dictionary.com, tesaurus.com y Wikipedia-api. Ejemplo: Manzana: usualmente redonda, roja o amarilla = manzana, redondo, rojo, amarillo

- Leer dos veces el conjunto de datos para eliminar entradas duplicadas y evitar inconsistencias al clasificar, ya que, en las instrucciones, el valor de 1 es para el caso de *palabra1* tiene relación con *atributo* y 0 para cualquier otro caso, esto incluye el caso específico de *palabra1* y *palabra2* tienen relación con el *atributo*, por lo que descartarlo reduce nuestra lista de palabras positivas y genera inconsistencias al clasificar.

4.6 Resolución de la tarea

Como lo mencionan los diferentes autores en capítulos anteriores la tarea de captura de atributos discriminativos depende del modelo y conjunto de datos que maneja para su resolución, por lo que la implementación de ciertos modelos, métodos y colecciones de datos existentes tienen un efecto positivo o negativo en la precisión de los resultados.

Como lo mencionan los trabajos de Escandell (2007) por la disciplina semántica y Faruqui et al. (2016) por la disciplina computacional la tarea de similitud de palabras tiene que analizar el significado semántico de la palabra y más allá, ya que este presenta diferentes problemas; hay que tomar en cuenta el lenguaje (idioma) y conjunto de datos (corpus, modelos, diccionarios) para predecir mejores resultados, por lo que los significados de una palabra son importantes pero puede ser complementados con relaciones semánticas para tener una amplia variedad de

resultados, asi como lo dice Tversky (1977) que la similitud debe describirse como una comparación de características, más que como una comparación de distancias métricas. Es esta la razón por la que el modelo propuesto trabaja sobre la combinación de recursos externos basados en significados de palabras (diccionarios) y vectores de palabras.

Al leer el conjunto de datos de entrenamiento se observó que la lista de ejemplos positivos fue separada de forma correcta, sin embargo la lista de ejemplos negativos contenía entradas duplicadas de la lista anterior, por lo que se tiene que tener en cuenta la descripción de la tarea en la cual indica que para cualquier combinación diferente de palabra1 es similar a un atributo se clasificaría con la etiqueta 0, lo que incluye alguna de las siguientes combinaciones de palabra1 no tiene relación con el atributo, ambas palabras no tienen relación con el atributo y ambas palabras tienen relación con el atributo, esta última combinación supone un problema de clasificación, ya que si se clasifica de manera incorrecta puede generar inconsistencias en los resultados finales, es por esa razón que se verifica la lista de ejemplos candidatos positivos y negativos dos veces, asi se eliminaran las entradas duplicadas e intentará clasificar las entradas positivas que se encuentren mal clasificadas, en caso de no poder clasificarlas se descartarán para evitar resultados erróneos.

Las funciones utilizan las herramientas anteriormente mencionadas, por la facilidad que proporcionan en la búsqueda de los diferentes elementos utilizados de la semántica léxica que incluye los siguientes puntos a destacar:

- Separar múltiples palabras o frases para solo trabajar con palabras
- Cambiar letras mayúsculas a minúsculas con el fin de mantener el mismo formato para todas las palabras
- Eliminar caracteres especiales y números de las palabras
- Utilizar diccionarios para agilizar la búsqueda de un elemento
- Raíz de una palabra, para evitar múltiples entradas de un mismo elemento
- Utilización de synsets
- Definición de la palabra (WordNet forma local)

- Sinónimos
- Hiperónimos (término general, ejemplo: árbol)
- Hipónimos (término específico de un hiperónimo, ejemplo: roble)
- Holónimo (un todo, ejemplo: pie)
- Merónimo (parte de, ejemplo: talón)
- Definición extensa de Wikipedia API
- Definición del significado actualizada (dictionary.com)
- Sinónimos y relaciones de palabras (thesaurus.com)
- Vectores de palabras (Spacy y sens2vec)

A continuación, se encuentra el algoritmo principal del modelo propuesto.

4.6.1 Algoritmo

A continuación, se muestra el algoritmo principal desarrollado de la tarea de atributos discriminativos; se explicará brevemente el comportamiento de cada instrucción dentro del algoritmo, posteriormente se describirá a detalle cada función desarrollada.

```
Inicio:
        dicPos={} //tipo diccionario
        dicNeg={}
        dicWeb={}
        dicPos,dicNeg=extraeDatos(training.txt) //extrae datos de entrenamiento en los diccionarios
        archivo=abrirArchivo(validation.txt)
        Para linea en archivo leerlinea:
        //linea es de la forma palabra1,palabra2,atributo
                linea=separar(',') //ahora es palabra1, palabra2, atributo
                w1aPos=(existe (palabra1,atributo) en dicPos) //verdad ó falso
                =(existe (palabra2,atributo) en dicPos)
                w1aNeg=(existe (palabra1,atributo) en dicNeg)
                w2aNeg=(existe (palabra2,atributo) en dicNeg)
                Si not(w1aPos): //w1a = falso
                        //Se utilizan librerías locales de diccionarios NLTK (ver función 3)
                        w1a=comparaPalabraAtributoL(palabra1,atributo)
                        //buscar la palabra con requests, wikipedia api y sense2vec
                        Si not(w1a) and not(existe (palabra1) en dicWeb):
                                w1a,features=comparaPalabraAtributoWeb(palabra1,atributo)
```

```
                        dicPos = actualizaDiccionario(features)
                        dicWeb[palabra1]=1 //fue buscado en web
                        Si not(w1a) and not(existe (atributo) en dicWeb):
                                w1a,features= comparaPalabraAtributoWeb (atributo,palabra1)
                                dicPos = actualizaDiccionario (features)
                                dicWeb[atributo]=1 //fue buscado en web
                        FinSi
                FinSi
        sino: //w1a = verdadero
                w1a=Verdadero
        FinSi
FinPara
Si not(w2aPos): //w2a = falso
        w2a= comparaPalabraAtributoL (palabra2,atributo)
        Si not(w2a) and not(existe (palabra2) en dicWeb):
                w2a,features= comparaPalabraAtributoWeb (palabra2,atributo)
                dicPos = actualizaDiccionario(features)
                dicWeb[palabra2]=1 //fue buscado en web
                Si not(w2a) and not(existe atributo en dicWeb):
                        w2a,features= comparaPalabraAtributoWeb (palabra2,atributo)
                        dicPos = actualizaDiccionario (features)
                        dicWeb[palabra2]=1 //fue buscado en web
                FinSi
        FinSi
sino:
        w2a=Verdadero
FinSi
Si ((w1aPos and w2aNeg) or (w1aPos and not(w2a)) or (w1a and w2aNeg) or (w1a and not(w2a)) ):
        imprimir("palabra1,palabra2,atributo,1") //Es atributo discriminativo
sino: //w1aNeg or (w1aPos and w2aPos)
        imprimir("palabra1,palabra2,atributo,0") //No es atributo discriminativo
        archivo cerrarArchivo()
FinSi
Fin
```

4.7 Métodos desarrollados

Los métodos desarrollados para el algoritmo están desarrollados con el lenguaje de programación Python v3.5 y las herramientas vistas anteriormente, a continuación, se describirán las funciones que fueron expuestas en la resolución de tareas, lo cual incluye entrada, salida de datos y funcionamiento.

4.7.1 synonym

La función 1 synonym a partir de una palabra encontrará sinónimos de la misma con los synsets de la herramienta de wordnet, los cuales serán truncados para solo almacenar el sinónimo, por otro las demás relaciones léxicas como hiperónimos, hipónimos, holónimos y merónimos, serán agregadas al diccionario, en caso de no encontrar resultados se omite.

Funciones que utiliza: allSynLemma una lista la indexa en un diccionario

Datos de entrada: palabra tipo string

Salida de datos: diccionario de datos (synset)

Ejemplo:

Entrada: sándwich

Donde cada Synset contiene la información de significado de la palabra, en este caso de tipo sustantivo (noun), con significado nos referimos a las relaciones léxicas como definición, sinónimos, hiperónimos, hipónimos, etc.

```
InicioFuncion synonym(word):
        synonyms={}
        palabras=synsets(word)
        Si longitud(palabras)==0:
                regresar synonyms
        FinSi
        //hypernyms
        lista=palabras.hypernyms()
        temp=allSynLemma(lista)
        Si longitud(temp)!=0:
                synonyms=actualizaDiccionario(temp)
        FinSi
        //hyponyms
        lista=palabras.hyponyms()
        temp=allSynLemma(lista)
        Si longitud(temp)!=0:
                synonyms=actualizaDiccionario(temp)
        FinSi
        //member_holonyms
        lista=palabras.member_holonyms()
        temp=allSynLemma(lista)
        Si longitud(temp)!=0:
                synonyms=actualizaDiccionario(temp)
        FinSi
        regresar synonyms
FinFuncion
```

Función 1

4.7.2 wordDefinition

La función 2 wordDefinition consulta en el diccionario de WordNet la definición de las palabras y extrae las palabras principales agregándolas a un diccionario, esto incluye separar la frase en palabras, eliminar palabras vacías, palabra raíz y cambiar todos los caracteres a minúsculas.

Funciones que utiliza: lemmatize_word extrae raíz de una palabra, separa frases, eliminar caracteres especiales y cambia caracteres a minúsculas.

Datos de entrada: palabra tipo string

Salida de datos: diccionario de datos

Ejemplo

Entrada: sandwich

Def: ['two (or more) slices of bread with a filling between them', 'make into a sandwich', 'insert or squeeze tightly between two people or objects']

```
InicioFuncion wordDefinition(word):
        dic={}
        def=list(word.definitions)
        Si longitud(def)!=0:
                lista=[]
                dic={}
                //Separar palabras por espacio ' '
                lista=separaPalabras(def)
                Para palabra en lista:
                        //lengua=stopwords del lenguaje
                        Si palabra not en lengua:
                                //Significado de una palabra
                                palabra=lemmatize_word(palabra)
                                //Agregar en diccionario
                                Si not(palabra en dic):
                                        dic[palabra]=1
                                FinSi
                        FinSi
                FinPara
        FinSi
        regresar dic
FinFuncion
```

Función 2

4.7.3 compareWorAtr

La función 3 compareWorAtr es una función importante que utiliza recursos léxicos de forma local, los cuales incluyen las funciones anteriormente mencionadas para almacenar sus características y ampliar su conjunto de datos.
Funciones que utiliza: lemmatize_word, wordDefinition, lemmalist y synonym
Datos de entrada: palabra# y atributo, dos palabras tipo string
Salida de datos: Verdadero o falso
Ejemplo: sándwich lunch
Def ['two (or more) slices of bread with a filling between them', 'make into a sandwich', 'insert or squeeze tightly between two people or objects']
Def ['the first meal of the day (usually in the morning)', 'eat an early morning meal', 'provide breakfast for']

```
InicioFuncion compareWorAtr(word,atri):
        def=list(word.definitions)
        lista=[]
        dic={}
        syn={}
        Para definition en def:
                //Separar palabras por espacio ' '
                lista=separaPalabras(def)
                Para palabra en lista:
                        //lengua=stopwords del lenguaje
                        Si palabra not en lengua:
                                //Significado de una palabra
                                palabra=lemmatize_word(palabra)
                                //Agregar en diccionario
                                Si not(palabra en dic):
                                        dic[palabra]=0
                                FinSi
                                //Significado de una palabra de la palabra (1)
                                dicDef=wordDefinition(palabra)
                                dic=actualizaDiccionario(dicDef)
                                //Lemma de la palabra anterior (2)
                                lemma=lemmalist(palabra)
                                dic=actualizaDiccionario(lemma)
                        FinSi
                FinPara
        FinPara
        //Sinonimo de la palabra principal (3)
        syn=synonym(word)
        dic=actualizaDiccionario(syn)
        regresar (atri en dic) //Verdadero o Falso
FinFuncion
```

Función 3

4.7.4 search_word_dictionary

La función 4 search_word_dictionary utiliza las herramientas de request y beautifulsoup para conseguir la información de dictionary.com puesto que contiene una amplia gama de definiciones actualizadas, ampliando el conjunto de listas de gran manera, además de brindar información actualizada al día de hoy. También se utiliza una estructura similar para la información de thesaurus.

Datos de entrada: palabra tipo string

Salida de datos: diccionario de datos

Ejemplo: sandwich

- Two or more slices of bread or the like with a layer of meat, fish, cheese, etc., between each pair.
- Open sandwich.
- Something resembling or suggesting a sandwich, as something in horizontal layers: a plywood sandwich.

```
InicioFuncion search_word_dictionary(search):
        dic={}
        results=requests("http://www.dictionary.com/browse/"+search)
        Si results != None:
                all_info=conseguirTexto(results)
                Para element en all_info:
                        word=separaPalabras(element)
                        Si longitud(words)>0:
                                dic=actualizaDiccionario(words_to_dictionary(search,words))
                        FinSi
                FinPara
        FinSi
        regresar dic
FinFuncion
```

Función 4

4.7.5 search_word_wiki

La función 5 search_word_wiki utiliza la herramienta de Wikipedia api la cual hace una consulta en sitio web de Wikipedia, el resultado es devuelto en texto plano separado por el formato de las secciones (asi como en el sitio web), de igual manera se realizan las acciones de separar las palabras principales, convertir caracteres a

minúsculas, eliminar palabras vacías y tomar en cuenta palabra raíz, con el fin de tener un conjunto de datos extenso y actualizado.
Funciones que utiliza: words_to_dictionary pasa de lista de palabras a diccionario
Datos de entrada: palabra tipo string
Salida de datos: diccionario de datos
Ejemplo: cucumber
Def: The cucumber is a creeping vine that roots in the ground and grows up trellises or other supporting frames, wrapping around supports with thin, spiraling tendrils. The plant may also root in a soilless medium and will sprawl along the ground if it does not have supports....

```
InicioFuncion search_word_wiki(search):
        dic={}
        page_py = wiki_wiki.page(search)
        Si page_py.exists():
                Para s en page_py.sections:
                        words=separaPalabras(s)
                        Si longitud(words)>0:
                                dic=actualizaDiccionario(words_to_dictionary(search,words))
                        FinSi
                FinPara
        FinSi
        regresar dic
FinFuncion
```

Función 5

4.7.6 search_word_sense2vec

La función 6 search_word_sense2vec crea vectores de la palabra y devuelve los más similares al contexto.
Funciones que utiliza: words_to_dictionary
Datos de entrada: palabra tipo string
Salida de datos: diccionario de datos
Ejemplo: lunch

- lunch
- dinner
- breakfast

- lunch_time
- lunchtime
- lunch/dinner
- quick_lunch
- late_lunch
- snack
- big_lunch

```
InicioFuncion search_word_sense2vec(search):
        wordsV=[]
        words,scores=[]
        dic={}
        try:
                words, scores = s2v_most_similar(search)
                Para word, score en (words, scores):
                        Si (score > 0.5):
                                wordsV=actualizaDiccionario(word)
                        FinSi
                FinPara
                dic=words_to_dictionary(search,wordsV)
                regresar dic
        except:
                regresar dic
FinFuncion
```

Función 6

El código completo está disponible en la sección de anexos.

Capítulo 5 Resultados y Discusión de Resultados

A continuación, se presentarán los resultados obtenidos del modelo propuesto, así como su análisis y comparación con cada uno de los modelos existentes, demostrando así que la tarea de atributos discriminativos es una buena solución para tareas de similitud, esto a través del uso de las características de un objeto.
Como mencionan Faruqui y Dyer (2014) la evaluación de similitud semántica entre palabras supone que existe una sola noción de similitud, por lo que utilizar algunos de los diferentes modelos existentes mostrará resultados diferentes independientemente del problema a resolver, de igual manera no existe un estándar para la representación de palabras en vectores, lo que tiene como consecuencia que en algunos conjuntos de datos resulte difícil encontrar diferencias significativas.
En la tabla 5.1 se puede ver la información del conjunto de datos de pruebas y validación con el que trabaja el modelo propuesto, así como las pruebas que se realizaron para los otros modelos existentes; el conjunto de datos de validación cuenta con 2722 líneas de ejemplos de las cuales 1364 son líneas positivas, lo que equivale a 50.11% de las líneas totales, el resto son líneas negativas.

Conjunto	Pruebas	%	Validación	%
Líneas Positivas	6591	37.07	**1364**	**50.11**
Líneas Negativas	11191	62.93	**1358**	**49.89**
Total	17782		**2722**	

Tabla 5.1 Conjunto de datos

Cabe hacer mención que el conjunto de datos de entrenamiento train tiene 17782 líneas de ejemplos, lo que es un buen número para el conjunto de entrenamiento, sin embargo, las líneas de ejemplos negativos son 11191 (62.93%), representando la mayoría de elementos del conjunto, además de que son generados aleatoriamente y contienen mucho ruido, ya que menciona Alicia Krebs en el planteamiento de la tarea es para tener un modelo rico en parámetros, lo que causa una dificultad para el conjunto de lista de ejemplos positivos, pues como ya

se mencionó, para el caso de que la etiqueta sea 0 y que ambas palabras tienen relación con el atributo, puede clasificarla de manera exitosa o simplemente descartarla.

Algunos de los modelos de hoy en día tienen sus medidas de similitud o relación por lo que se pusieron a prueba con el modelo de entrenamiento anterior, los resultados se muestran en la tabla 5.2. Las pruebas realizadas incluyen:

- NLTK
  - Path length (path)
  - Leacock y Chodorow (lch)
  - Wu y Palme (wup)
  - Resnik (res)
- Relaciones de semejanza (RS)
  - Sinónimos (syn)
  - Hiperónimos (hype)
  - Hipónimos (hypo)
  - Holónimos y meronimos (holo)
  - Todos los anteriores (All)
- Spacy similarity
- Sense2vec
  - 300 ejemplos
  - 900 ejemplos

La comparación de éstos métodos demuestra y confirma lo que mencionaron los autores, puesto que para las medidas de similitud Path length (path), Leacock y Chodorow (lch) y Wu y Palme (wup) se obtuvo una precisión de 49.88 a 50.33 donde apenas es la mitad del conjunto de datos, por otra parte el método Resnik (res) hace uso de un documento de entrenamiento externo (además de los datos de entrenamiento) demostrando así que los resultados para este tipo de métodos dependen en gran medida del documento proporcionado, el cual tiene que especializarse en un tema en concreto. De la misma manera los métodos de vectores de palabras dependen de un conjunto de datos, pues como ya se mencionó ningún modelo maneja un estándar, esto lo podemos ver para los métodos de

sense2vec y spacy, ambos son modelos de vectores de palabras, pero utilizan diferentes medidas de similitud, sus resultados son un poco más favorables con una precisión de 50.84 a 53.41 de porcentaje positivo; sin embargo sus diccionarios locales son de un tamaño considerable, lo que se refleja en tiempo de ejecución.

En las relaciones de semejanza como sinónimos (syn), hiperónimos (hype), hipónimos (hypo), holónimos y merónimos (holo) se obtienen resultados con una precisión de 54.44 a 54.51 porciento de aciertos, lo que en conjunto con el significado de la palabra resulta en 55.51 porciento de precisión.

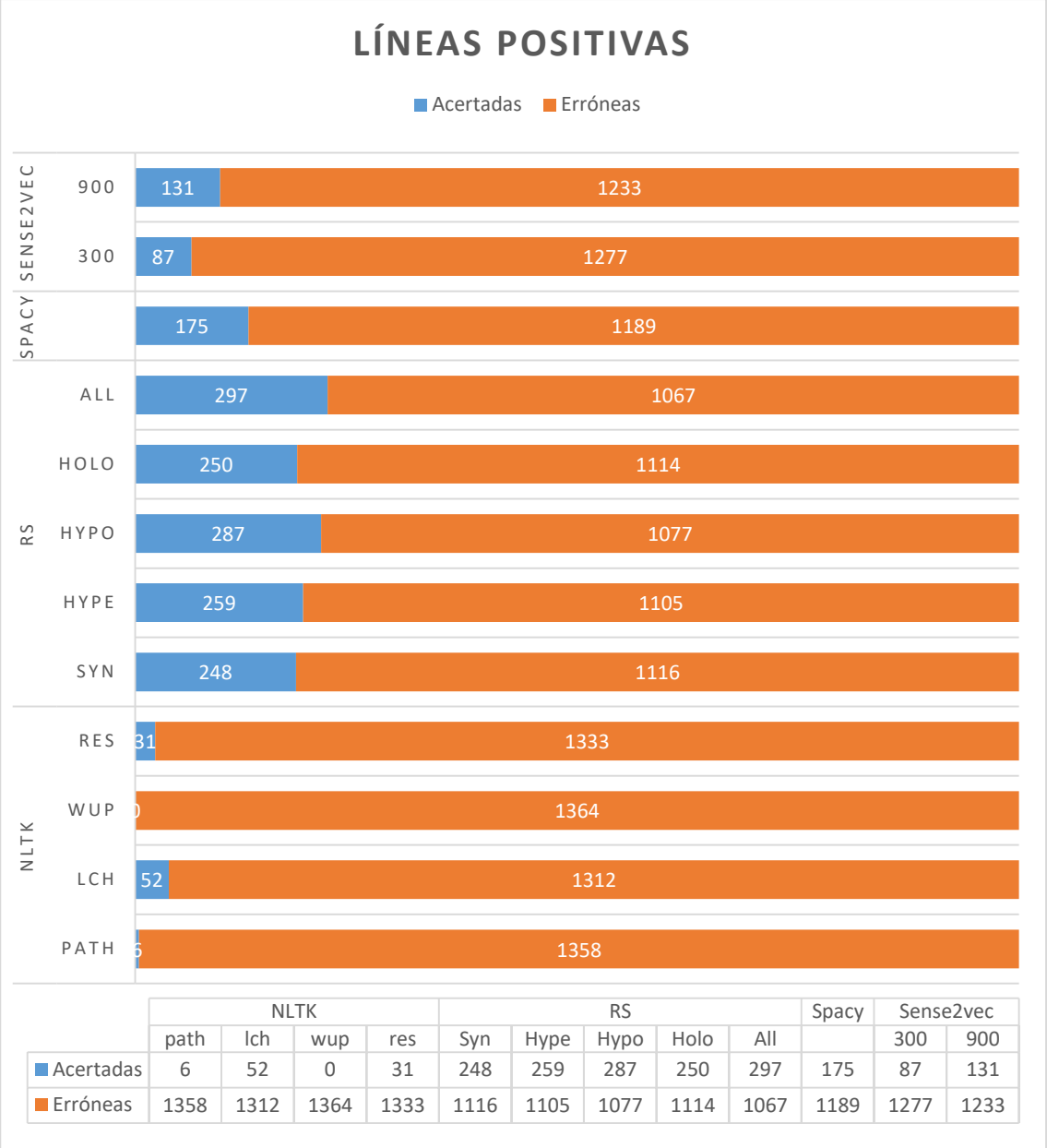
	NLTK				**RS**				
	path	**lch**	**wup**	**res**	**Syn**	**Hype**	**Hypo**	**Holo**	**All**
Diccionarios									
Positivo	824	889	817	864	1043	1063	1080	1046	1100
Negativo	14563	14638	14556	14600	14875	14888	14922	14877	14932
Líneas Positivas									
Acertadas	6	52	0	31	248	259	287	250	297
Erróneas	1358	1312	1364	1333	1116	1105	1077	1114	1067
Porcentaje	**0.43**	**3.81**	**0**	**2.27**	**18.18**	**18.98**	**21.04**	**18.32**	**21.77**
Líneas Negativas									
Acertadas	1357	1318	1358	1341	1234	1224	1222	1234	1214
Erróneas	1	40	0	17	124	134	136	124	144
Porcentaje	**99.92**	**97.05**	**100**	**98.74**	**90.86**	**90.13**	**89.98**	**90.86**	**89.39**
Total									
Líneas Acertadas	1363	1370	1358	1372	1482	1483	1509	1484	**1511**
Líneas Erróneas	1359	1352	1364	1350	1240	1239	1213	1238	**1211**
Porcentaje total	**50.07**	**50.33**	**49.88**	**50.40**	**54.44**	**54.48**	**55.43**	**54.51**	**55.51**

	Spacy	**Sense2vec**	
		300	**900**
Diccionarios			
Positivo	1018	340938	1003401
Negativo	14791	14695	14758
Líneas Positivas			
Acertadas	175	87	131
Erróneas	1189	1277	1233
Porcentaje	**12.82**	**6.37**	**9.60**
Líneas Negativas			
Acertadas	1279	1297	1275
Erróneas	79	61	83
Porcentaje	**94.18**	**95.50**	**93.88**
Total			
Líneas Acertadas	1454	1384	1406
Líneas Erróneas	1268	1338	1316
Porcentaje total	**53.41**	**50.84**	**51.65**

Tabla 5.2 Comparativa de modelos

En las siguientes Figuras se muestran gráficas con los resultados obtenidos de líneas positivas (Figura 5.1), líneas negativas (Figura 5.2) y total de líneas acertadas/erróneas (Figura 5.3), donde el color azul son las líneas acertadas, el color naranja las líneas erróneas y debajo de la gráfica se encuentran los datos exactos de cada modelo; esto con el fin de observar de manera más clara la comparativa entre los modelos existentes.

Las líneas positivas muestran mejores resultados para los modelos de similitud de semejanza, los cuales utilizan similitud semántica léxica, esto se puede ver de manera más clara en la Figura 5.1, por detrás de estos se encuentran los modelos de similitud de palabras, pero como ya se ha mencionado, estos modelos utilizan el vector de la palabra más cercano (de acuerdo a su conjunto de datos), por lo que puede clasificar de manera errónea varios elementos.

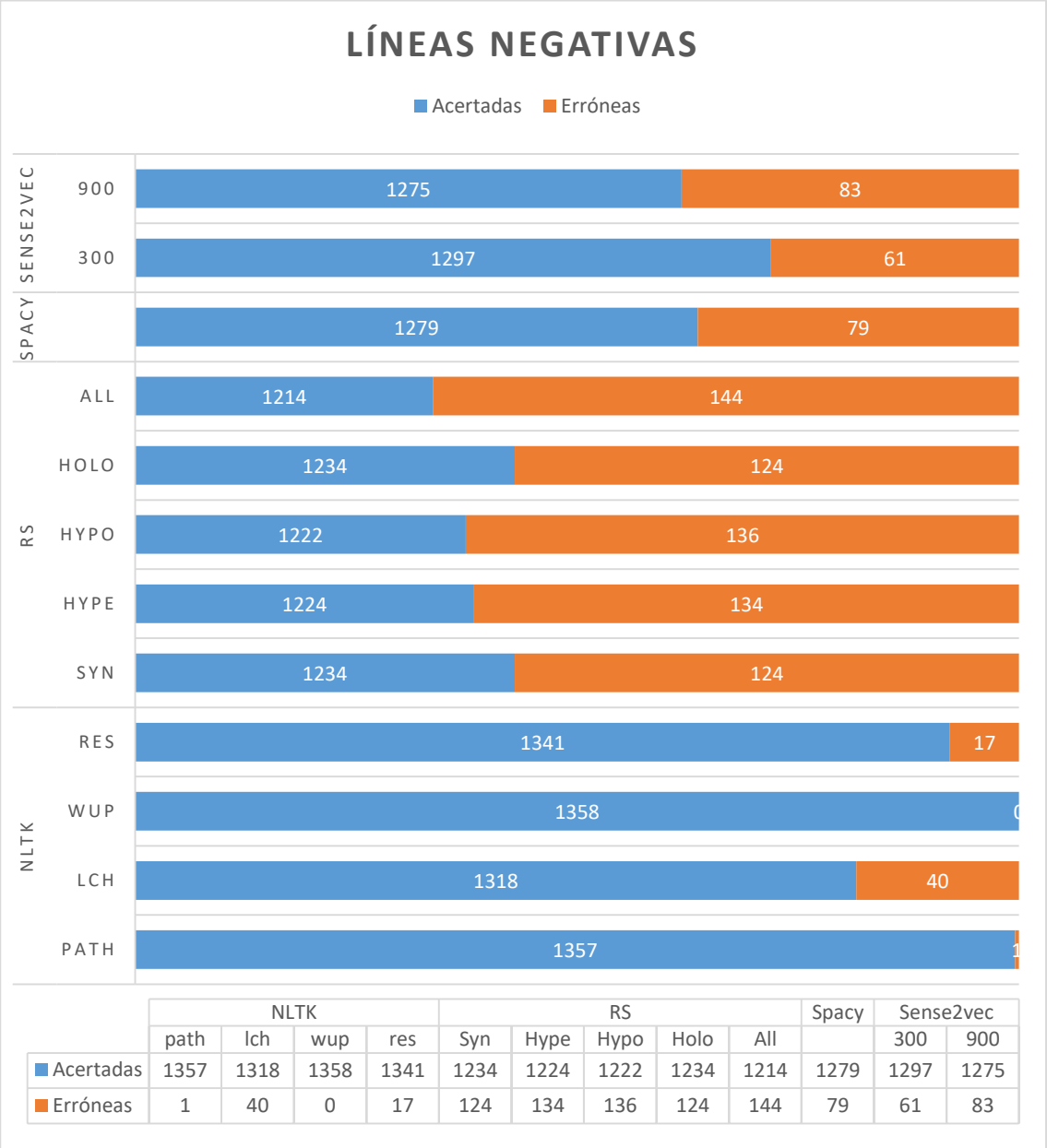


	NLTK				RS					Spacy	Sense2vec	
	path	lch	wup	res	Syn	Hype	Hypo	Holo	All		300	900
Acertadas	6	52	0	31	248	259	287	250	297	175	87	131
Erróneas	1358	1312	1364	1333	1116	1105	1077	1114	1067	1189	1277	1233

Figura 5.1 Gráfica líneas positivas

En la Figura 5.2 se puede observar los resultados de cada modelo para la clasificación de líneas negativas, donde están los casos donde ambas palabras tienen relación con el atributo, ninguna palabra tiene relación con el atributo y cualquier caso donde no se pueda clasificar, lo que indica que los modelos de similitud por formula (NLTK) muestran mejores resultados, sin embargo, tomando

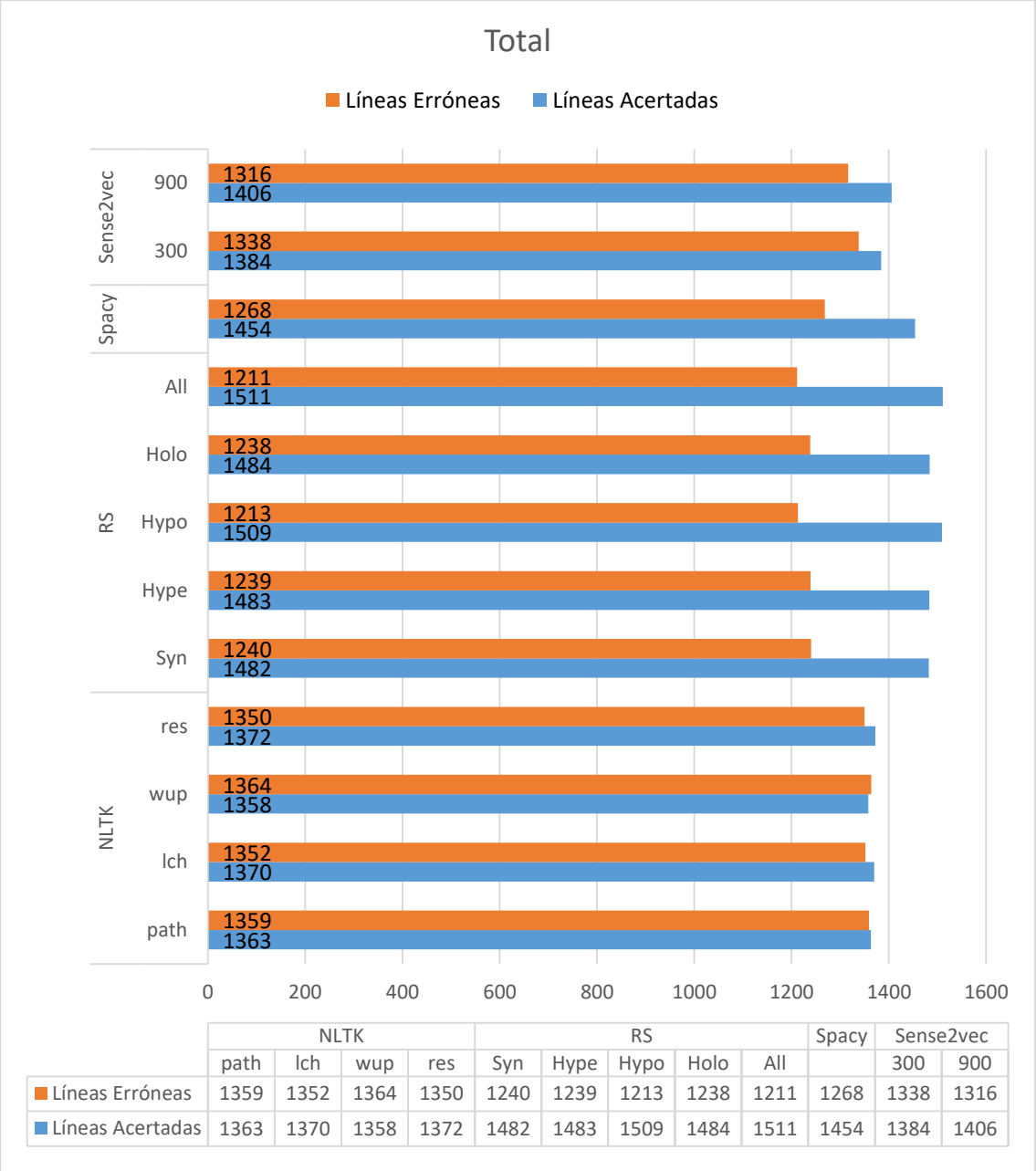
en cuenta los resultados anteriores es porque no pueden clasificar la mayoría de elementos.



	NLTK				RS					Spacy	Sense2vec	
	path	lch	wup	res	Syn	Hype	Hypo	Holo	All		300	900
Acertadas	1357	1318	1358	1341	1234	1224	1222	1234	1214	1279	1297	1275
Erróneas	1	40	0	17	124	134	136	124	144	79	61	83

Figura 5.2 Gráfica líneas negativas

En la Figura 5.3 se muestran los resultados de líneas de ejemplos positivos como negativos, recordar la tabla 5.1 donde la cantidad de líneas positivas es 1364 y líneas negativas 1358.



	NLTK				RS					Spacy	Sense2vec	
	path	lch	wup	res	Syn	Hype	Hypo	Holo	All		300	900
Líneas Erróneas	1359	1352	1364	1350	1240	1239	1213	1238	1211	1268	1338	1316
Líneas Acertadas	1363	1370	1358	1372	1482	1483	1509	1484	1511	1454	1384	1406

Figura 5.3 Gráfica total líneas

Algunos aspectos importantes que destacan de los resultados obtenidos es el tamaño del diccionario de cada método, pues como se observa en la tabla 6, este depende en gran medida del tipo de método utilizado, pues en los métodos con menor porcentaje no cambia en gran medida con respecto al original, además para

los métodos de vectores de palabras los elementos del diccionario crecen exponencialmente, lo que resulta en tener una gran colección de palabras, pero como se observa en las relaciones de semejanza solo se extrae la semántica léxica de los métodos mencionados anteriormente de la palabra a buscar, lo que proporciona palabras clave a nivel semántica para identificar la palabra objetivo, sin embargo no hay que dejar de lado los métodos de vectores de palabras pues como se mencionó estos no son estándar y su tiempo de ejecución es menor en comparación con los demás. Otro punto a destacar es que los porcentajes de líneas positivas entre mayor sea, mejora la precisión del resultado, pero al mismo tiempo baja el porcentaje de líneas negativas, esto se puede ver claramente en las filas de porcentaje para líneas positivas, negativas y total de la tabla 5.2.

El modelo propuesto implementa los métodos mencionados anteriormente, además de integrar herramientas en línea como Wikipedia api, dictionary.com y thesaurus.com, para ampliar el conjunto de datos con todos los elementos posibles y actualizados al día de hoy. Los resultados del modelo propuesto son mostrados a continuación (tabla 5.3).

Modelo	
Diccionarios	
Positivo	1384979
Negativo	15227
Líneas Positivas	
Acertadas	577
Erróneas	787
Porcentaje	**42.30**
Líneas Negativas	
Acertadas	1171
Erróneas	187
Porcentaje	**82.22**
Total	
Líneas Acertadas	1748
Líneas Erróneas	974
Porcentaje total	**64.21**

Tabla 5.3 Resultados del modelo propuesto

En los resultados de la tabla 5.3 los diccionarios finales son de gran tamaño comparados con los anteriores, ya que esta combina todos los métodos anteriormente mencionados; al incrementar el diccionario de líneas positivas con los elementos de semántica léxica y vectores de palabras este proporciona un extenso conjunto de datos, dando como consecuencia encontrar más coincidencias, sin embargo, características de un objeto o entidad muy específicos no son encontrados en las herramientas utilizadas.

Krebs y Paperno (2016) describen que un sistema para la comprensión básica del lenguaje debería ser capaz de detectar cuándo los conceptos son similares entre sí, pero también de qué manera los conceptos difieren. Además de mencionar que en el futuro, resolver la tarea de similitud podría ayudar en diversas aplicaciones, por ejemplo, lexicografía automatizada (generación automática de características para incluir en definiciones de diccionario), agentes de conversión (la elección de elementos léxicos con características diferenciales puede ayudar a crear diálogos más apropiados, similares a los humanos), traducción automática (donde explícitamente tiene en cuenta las diferencias semánticas entre las variantes de traducción puede mejorar la calidad de la salida), etc. Con esto nos demuestran que resolver el problema de diferencia puede tener grandes aportaciones al desarrollo de procesamiento de lenguaje natural.

Capítulo 6 Conclusiones

En este trabajo se investigaron las técnicas, fórmulas, modelos y conjuntos de datos que se utilizan en tareas de similitud entre pares de palabras o frases cortas; presentando una introducción al área de la semántica léxica y Procesamiento de Lenguaje Natural, mostrando así los diferentes términos que utiliza la semántica en el lenguaje natural, asi como los principales obstáculos que presenta esta tarea a nivel semántico, pues una lengua siempre está en constante desarrollo, lo que provoca que no se pueda formalizar de manera general; de igual manera se revisaron algunas fórmulas de similitud como son formula de coseno, Resnik, Wu y Palmer, entre otros, de tal manera que para validar la investigación se sometieron a comparación algunos de los modelos mencionados anteriormente, dicha comparativa consiste en entrenar los modelos con el mismo conjunto de datos para que entreguen resultados con el mismo archivo de pruebas.

Los resultados obtenidos de esta comparativa confirman la investigación realizada, además de ayudar al desarrollo del modelo propuesto pues al tomar en cuenta los problemas, recomendaciones y trabajos de los autores mencionados, los resultados mejoran a los modelos comparados, demostrando de esta manera que para diferenciar y definir a un objeto o entidad con medidas de similitud es recomendable tener en cuenta algunos aspectos principales, el primero es tomar los problemas de semántica léxica dentro del lenguaje natural pues este depende de la lengua y zona geográfica a la que se hace referencia, también depende del contexto de la situación, dado que algunas palabras pueden tener definiciones completamente diferentes, un claro ejemplo de lo anterior es la palabra arco (tiene significado de arma, línea, etc.), por el lado de PLN los problemas de semántica léxica son los mismos, además de que los modelos convencionales tienen problemas de clasificación de similitud pues estos ocupan métodos que calculan la similitud y relación con otras palabras pero no logran descubrir las diferencias significativas, lo que conduce a resultados erróneos o diferentes de los esperados por manejar diferentes conjuntos de datos. Otro aspecto importante es hacer uso de los atributos discriminativos que definen al objeto, pues sus características

principales van a diferenciar similitud de relación que tiene con otro objeto en común.

Hallamos que los métodos de similitud por medio de una fórmula no obtienen buenos resultados ya que no pueden clasificar la mayoría de los elementos, los modelos a partir de vectores de palabras multidimensionales y de contexto mostraron una precisión de un poco más de la mitad de aciertos, sin embargo, los resultados dependen del conjunto de datos que utiliza. Al utilizar las relaciones de semejanza se observaron mejores resultados en la precisión de líneas positivas y mayor puntaje en líneas acertadas, debido a que se compara su objeto de entrada a nivel de semántica léxica, es decir que la palabra es definida a partir de otra palabra la cual tiene un significado dentro de un diccionario y este puede definir a los objetos a partir de una definición, sinónimos de la palabra, partes de un todo, etc.

El modelo propuesto hace uso de estas relaciones de semejanza con la ayuda de las librerías de Wordnet, NLTK, Wikipedia Api, dictionary, thesaurus y sense2vec para extraer las principales características de un objeto a nivel semántica y de vectores de palabras, entregando los resultados de 64.21 de precisión para el conjunto de entrenamiento. En la participación de la tarea 10 de SemEval 2018 con el conjunto de pruebas (test) se obtuvo 0.63/1.0 de precisión, dicho conjunto no tiene etiquetas de clasificación y para la evaluación de los resultados es necesario enviar los resultados del conjunto de datos por el modelo propuesto a la página en línea de CodaLab[11].

Por último, la investigación presentada y el modelo propuesto para esta tesis constituyen una evidencia de que la tarea de atributos discriminativos es una propuesta de solución válida al problema de similitud entre pares de palabras, el modelo permite extraer las características principales de un objeto a nivel semántico y con ayuda de las herramientas mencionadas este conjunto de datos que obtiene es lo más actualizado posible, resultando así la generación de características de un objeto a partir de su semántica léxica, un aporte para futuras investigaciones.

[11] https://competitions.codalab.org/competitions/17326#results

Capítulo 7 Anexos

7.1Instalación de software

Requiere Python 3.5

Instalar las siguientes librerías:

Nltk3.4

- sudo pip3 install -U nltk
- Corpus
  - python3
  - import nltk
  - nltk.download()

Wordnet

Textblob 0.15.2

- pip3 install -U textblob
- python3 -m textblob.download_corpora

SpacyV2.0.18

- pip3 install -U spacy
- python3 -m spacy download en
- python3 -m spacy download en_vectors_web_lg

Request V stable

- pip3 install requests

Beautiful Soup V4.4.0

- pip3 install bs4

Wikipedia-API V0.4.0

- pip3 install wikipedia-api

7.2 Código fuente

```
# -*- coding: utf-8 -*-
"""
@author: AlbertoTP
"""

import time
import sys
import re
import requests

import spacy
#nlp = spacy.load('en')
nlp = spacy.load('en_vectors_web_lg')
from spacy.lemmatizer import Lemmatizer
from spacy.lang.en import LEMMA_INDEX, LEMMA_EXC, LEMMA_RULES

from textblob import Word

from nltk.stem import WordNetLemmatizer
from nltk.corpus import wordnet as wn
#Stopwords in English
from nltk.corpus import stopwords as sw
en_sw = set(sw.words('english'))

from bs4 import BeautifulSoup

#wikipedia in english
import wikipediaapi
wiki_wiki = wikipediaapi.Wikipedia('en')

#Sense2vec
import sense2vec
s2v = sense2vec.load('/usr/local/lib/python3.5/dist-packages/sense2vec/reddit_vectors-1.1.0/')


def readFile(ruta,diccPos,diccNeg):
    """
    Read file, extract information (Word 1, word 2, Feature, value) line per line, insert in
dictionaries
    Input: ruta path of the file to be read (String)
            diccPos dictionary of candidate positive examples (dictionary)
            diccNeg dictionary of candidate negative examples (dictionary)
    Return: 2 dictionaries diccPos and diccNeg (dictionary)
    """
    print ("\n>>> Archivo(train): ",ruta," extrayendo datos ...")
    #two times, one for fill dictionaries and other for delete duplicate key
    for i in range (0,2):
        try:
            file = open(ruta,"r")
        except IOError:
            print ("There was an ERROR reading file")
            sys.exit()
        """
        Data format: 4 comma-separated fields:
        - word 1 (pivot)
        - word 2 (comparison)
        - feature
        - label ("1" if the feature characterizes word 1 compared to word 2, "0" otherwise)
        Example:    word1,word2,feature,label(0/1)
        """
        linePos,lineNeg=0,0
        for linea in file.readlines():
            line=linea.split(",")
            if line[3]=='1\n': #agrega a positivos
```

```
                linePos+=1
                #if diccPos.has_key((line[0],line[2])):
                if ((line[0],line[2]) in diccPos):#python3
                    diccPos[line[0],line[2]].append(line[1])
                else:
                    diccPos[line[0],line[2]]=[line[1]]
                #if diccNeg.has_key((line[1],line[2])):
                if ((line[1],line[2]) in diccNeg):#python3
                    diccNeg[line[1],line[2]].append(line[0])
                else:
                    diccNeg[line[1],line[2]]=[line[0]]
            else:  #agrega a negativos
                lineNeg+=1
                #if not(diccPos.has_key((line[0],line[2]))):
                if not((line[0],line[2]) in diccPos):#python3
                    #if diccNeg.has_key((line[0],line[2])):
                    if ((line[0],line[2]) in diccNeg):#python3
                        diccNeg[line[0],line[2]].append(line[1])
                    else:
                        diccNeg[line[0],line[2]]=[line[1]]
                else:
                    #if not(diccPos.has_key((line[1],line[2]))):
                    if not((line[1],line[2]) in diccPos):#python3
                        diccPos[line[1],line[2]]=[line[0]]

        #Delete duplicate keys in dictionay negative
        #cont=0
        for key in diccPos:
            #if diccNeg.has_key(key):
            if (key in diccNeg):#python3
                del diccNeg[key]
                #cont+=1
        #print "Se descartaron ",cont," palabras"
        file.close()
    print ("lineas Pos: ",linePos,"\nlineas Neg: ",lineNeg,"\n")
    return diccPos,diccNeg

def lemmalist(word):
    """
    Lemmatisation is the process of grouping together the inflected forms of a word so they can be
analysed as a single item, identified by the word's lemma, or dictionary form
    Input: word (string)
    Return: lemma (dictionary)
    """
    syn_set = {}
    #print "lemmasWord",word
    cont=0
    for synset in wn.synsets(word):
        if cont<1:
            for item in synset.lemma_names():
                if not(item in syn_set):
                    syn_set[item]=2;
            cont+=1
        else:
            break
    #print syn_set
    return syn_set



def allSynLemma(lista):
    """
    Separates a list and insert each word in a dictionary, also it separtes compoun word (word_1 =
word,1)
    Input: lista (list)
    Return: synonims (dictionary)
    """
    synonyms={}
    if len(lista)!=0:
```

```
        for element in lista:
            #print ">>> ", element.lemma_names()
            for i in element.lemma_names():
                #print i
                vec=i.split("_")
                if len(vec)>1:
                    for j in vec:
                        #if not(synonyms.has_key(j)):
                        if not(j in synonyms):#python3
                            synonyms[j]=3
                #elif not(synonyms.has_key(i)):
                elif not(i in synonyms):#python3
                    synonyms[i]=3;
    return synonyms

def synonym(word):
    """
    first synonyms, hypernyms, hyponyms and member_holonyms of the one word
    Input: word (String)
    Return: synonims (dictionary)
    """
    synonyms={}
    #print ">>>def synonym:", str(wn.morphy(word))
    palabras=wn.synsets(str(wn.morphy(word)))
    #print palabras
    if len(palabras)==0:
        return synonyms

    temp=str(palabras[0])
    temp=temp[8:-2]
    palabras = wn.synset(str(temp))

    #hypernyms
    lista=palabras.hypernyms()
    temp=allSynLemma(lista)
    if len(temp)!=0:
        synonyms.update(temp)

    #hyponyms
    lista=palabras.hyponyms()
    temp=allSynLemma(lista)
    if len(temp)!=0:
        synonyms.update(temp)

    #member_holonyms
    lista=palabras.member_holonyms()
    temp=allSynLemma(lista)
    if len(temp)!=0:
        synonyms.update(temp)
    return synonyms

def wordDefinition(word):
    """
    The meaning of a word according to wordnet, remove stopwords
    Input: word (String)
    Return: (dictionary)
    """
    dic={}
    definitions=list(Word(word).definitions)
    #print definitions
    if len(definitions)!=0:
        lista=[]
        dic={}
        lista=definitions[0].split(' ')
        for palabra in lista:
            if palabra not in en_sw:
                #Meaning of a word
                palabra=lemmatize_word(palabra)
                #Insert in the dictionary
```

```
                if not(palabra in dic):
                    dic[palabra]=1
    return dic

def lemmatize_word(word):
    """
    Assign the base form of words
    Input: word (plural, capital letters)
    return: word (singular, lower letters)
    """
    palabra=word
    palabra=Word(str(palabra).lower() )
    palabra=palabra.lemmatize()
    if palabra==word:
        lemmatizer = Lemmatizer(LEMMA_INDEX, LEMMA_EXC, LEMMA_RULES)
        palabra = lemmatizer(word, u'NOUN')
        palabra=palabra[0]
    return palabra

def compareWorAtr(word,atri):
    """
    It extracts all the characteristics (Atributes) of a word according to the meaning of the
WordNet dictionary
    remove stopwords and add synonyms (update dictionary with other dictionary)
    Input: word, atribut (string)
    return: (boolean)
    """
    definitions=list(Word(word).definitions)
    lista=[]
    dic={}
    syn={}
    #lemma={}
    for definition in definitions:
        lista=definition.split(' ')
        for palabra in lista:
            if palabra not in en_sw:

                #Meaning of a word (0)
                palabra=lemmatize_word(palabra)
                #Insert in the dictionary
                if not(palabra in dic):
                    dic[palabra]=0

                #Meaning of a word of the word (1)
                dicDef=wordDefinition(str(palabra))
                dic.update(dicDef)

                #Lemma of the word of the word (2)
                lemma=lemmalist(palabra)
                #print lemma
                dic.update(lemma)

    #Synonyms of the main word (3)
    syn=synonym(str(word).lower())
    #Update main dictionary
    dic.update(syn)
    return (atri in dic,dic)

def words_to_dictionary(search,words):
    """
    delete special characters, numbers and stopwords. lemmatize the word and insert in a dictionary
    Input: words (Vector String)
    Return: dic (dictionary)
    """
    dic={}
    for word in words:
        #word=str(word.encode('ascii', 'ignore')).lower()
        word=str(word.lower())
        word=re.sub(r'[0-9]', '', word)
```

```
        word=re.sub('\W+','', word )
        if word not in en_sw:
            wordnet_lemmatizer = WordNetLemmatizer()
            word=wordnet_lemmatizer.lemmatize(word)
            #if not( dic.has_key(word) ): #python2
            if not( word in dic ): #python3
                dic[(search,word)]=["R"]
    return dic


def search_word_dictionary(search):
    """
    Search a word in dictionary.com
    Input: search (string)
    Return: dic (dictionary)
    """
    dic={}
    r=requests.get("http://www.dictionary.com/browse/"+search)

    soup=BeautifulSoup(r.text, "html.parser")
    results=soup.find("div",{"class":"css-8lgfcg e1iplpfw1"})
    if results != None:
        all_info=results.findAll("ol",{"css-zw8qdz e1hk9ate4"})
        for item in all_info:
            line=item.findAll("li",{"class":"css-2oywg7 e1q3nk1v3"})
            for element in line:
                cad=element.get_text()
                words=cad.split(' ')
                if len(words)>0:
                    dic.update(words_to_dictionary(search,words))
    return dic

def search_word_thesaurus(search):
    """
    Search a word in thesaurus.com
    Input: search (string)
    Return: dic (dictionary)
    """
    dic={}
    r=requests.get("https://www.thesaurus.com/browse/"+search)

    soup=BeautifulSoup(r.text, "html.parser")
    results=soup.find("section",{"class":"MainContentContainer css-7vy5fk e1h3b0ep0"})
    if results != None:
        all_info=results.findAll("ul",{"css-1lc0dpe et6tpn80"})
        for item in all_info:
            line=item.findAll("li")
            for element in line:
                cad=element.get_text()
                words=cad.split(' ')
                if len(words)>0:
                    dic.update(words_to_dictionary(search,words))
    return dic

def search_word_wiki(search):
    """
    Search a word in en.wikipedia.org
    Input: search (string)
    Return: dic (dictionary)
    """
    dic={}
    page_py = wiki_wiki.page(search)
    if page_py.exists():
        level=0
        for s in page_py.sections:
            cad=s.text
            words=cad.split(' ')
            if len(words)>0:
                dic.update(words_to_dictionary(search,words))
```

```
            level+=1
    return dic

def search_word_sense2vec(search):
    """
    Search a word with sense2vec to return the most similar words of the search
    Input: search (string)
    Return: dic (dictionary)
    """
    wsearch=search+"|NOUN"
    wordsV=[]
    dic={}
    try:
        freq, vector = s2v[wsearch]
        words, scores = s2v.most_similar(vector, n=900)
        for word, score in zip(words, scores):
            if (score > 0.5):
                temp=word[0:-5]
                wordsV.append(temp)
        dic=words_to_dictionary(search,wordsV)
        return dic
    except:
        return dic

def compare_word_feature(word,atri):
    """
    It extracts all the characteristics (Atributes) of a word according to the meaning of the online
dictionary, wiki and sense2vec
    Input: word, atribut (string)
    return: true or false (boolean)
    retunr: dictionary with word,atrib (dictionary)
    """
    features={}
    features=search_word_dictionary(word)
    features.update(search_word_thesaurus(word))
    features.update(search_word_wiki(word))
    features.update(search_word_sense2vec(word))

    #print "len:",len(features),"\nDicc:",features
    #return features.has_key((word,atri)),features
    return ((word,atri) in features),features#python3

def similaritySpyCy(word1,word2):
    """
    Similarity between two words with Spicy
    Input: word1, word2 (String)
    Return: similarity (float)
    """
    tokens = nlp( (word1+" "+word2))
    #print tokens
    #print "Similarity Spacy:",tokens[0].similarity(tokens[1])
    try:
        if tokens[0].similarity(tokens[1])>0.5:
            return True
    except:
        return False
    return False

def main():
    starting_point = time.time()
    dicPos={}
    dicNeg={}
    dicWeb={}
    dicPos,dicNeg=readFile("dataTrain/train.txt",dicPos,dicNeg) #diccionarios del train
    #dicPos1,dicNeg1=readFile("dataTrain/validation.txt",dicPos,dicNeg)
    #dicPos.update(dicPos1)
    #dicNeg.update(dicNeg1)
    print ("Diccionarios Train")
    print ("elementos dicPos ",len(dicPos))
```

```
print ("elementos dicNeg ",len(dicNeg))
print ("Espere ...")

path="dataTrain/validation.txt"
#path="dataTest/test_triples.txt" #original file
ruta="resultados-M1.3.0.txt"
try:
    file = open(path,"r")
    result=open(ruta,"w")
except IOError:
    print ("There was an ERROR reading file")
    sys.exit()

archivo=[]
for linea in file.readlines():
    line=linea.rstrip('\n').split(",")
    #line=tuple(line)
    #print type(line)
    #print line #('word1', 'word2', 'atrib')
    #print line[0]," ",line[2]
    archivo.append(line)
file.close()
print ("\n>>> Archivo(test): ",path)
print ("Numero de lineas",len(archivo))

print ("\nClasificando (esto puede tomar tiempo) ...\n")

for line in archivo:
    temp0=line[0]
    temp1=line[1]
    temp2=line[2]
    line[0]=lemmatize_word(line[0])
    line[1]=lemmatize_word(line[1])
    line[2]=lemmatize_word(line[2])

    w1aPos=((line[0],line[2]) in dicPos)
    w2aPos=((line[1],line[2]) in dicPos)
    w1aNeg=((line[0],line[2]) in dicNeg)
    w2aNeg=((line[1],line[2]) in dicNeg)

    #Find the meaning of the word in a dictionary
    if not(w1aPos):
        w1a,features=compareWorAtr(line[0],line[2])
        dicPos.update(features)
        #Search the word in WEB and update the variable and the dictionary
        if not(w1a) and not((line[0]) in dicWeb):
            w1a,features=compare_word_feature(line[0],line[2])
            dicPos.update(features)
            dicWeb[line[0]]=1
            if not(w1a) and not((line[2]) in dicWeb):
                w1a,features=compare_word_feature(line[2],line[0])
                dicPos.update(features)
                dicWeb[line[2]]=1
                if not(w1a):
                    w1a=similaritySpyCy(line[0],line[2])
    else:
        w1a=True

    if not(w2aPos):
        w2a,features=compareWorAtr(line[1],line[2])
        dicPos.update(features)
        #Search the word in WEB and update the variable and the dictionary
        if not(w2a) and not((line[1]) in dicWeb):
            w2a,features=compare_word_feature(line[1],line[2])
            dicPos.update(features)
            dicWeb[line[1]]=1
            if not(w2a) and not(line[2] in dicWeb):
                w2a,features=compare_word_feature(line[1],line[2])
                dicPos.update(features)
```

```
                    dicWeb[line[1]]=1
                    if not(w2a):
                        w2a=similaritySpyCy(line[1],line[2])
        else:
            w2a=True

        if w1aNeg or (w1aPos and w2aPos): #dictionary negative
            #dicNeg[line[0],line[2]].append(line[1])
            result.write(str(temp0)+","+str(temp1)+","+str(temp2)+",0\n");
        elif ((w1aPos and w2aNeg) or (w1aPos and not(w2a)) or (w1a and w2aNeg) or (w1a and not(w2a))
):
            #dictionary positive
            if not(w1aPos): #inserta w1 en dicPos
                dicPos[line[0],line[2]]=[line[1]]
            if not(w2aNeg): ##inserta w2 en dicNeg
                dicNeg[line[1],line[2]]=[line[0]]
            result.write(str(temp0)+","+str(temp1)+","+str(temp2)+",1\n");
        else: #dictionary negative
            #dicNeg[line[0],line[2]]=[line[1]]
            result.write(str(temp0)+","+str(temp1)+","+str(temp2)+",0\n");

    result.close()

    print ("\nDiccionarios Train y Test despues de clasificar")
    print ("elementos dicPos ",len(dicPos))
    print ("elementos dicNeg ",len(dicNeg))

    #print dicPos.keys()
    print ("\nPuede revisar el archivo de resultados en: ",ruta)
    print ("Execution Time: ",time.time()-starting_point)

if __name__ == "__main__":
    main()
```

También disponible en github[12]

Para ejecutar el programa

python3 model_v1.3.0.py

[12] https://github.com/AlbertoTP/CapturaAtributosDiscriminativos

Capítulo 8 Referencias

1. Álvarez Carmona Miguel Ángel (2014). Detección de similitud en textos cortos considerando traslape, orden y relación semántica de palabras (Tesis de maestría). Instituto Nacional de Astrofísica, Óptica y Electrónica, Tonantzintla, Puebla.
2. Baroni Marco and Dinu Georgiana and Kruszewski Germán. (2014a). Don't count, predict! A systematic comparison of context-counting vs. context-predicting semantic vectors. Association for Computational Linguistics Sitio web: http://aclweb.org/anthology/P14-1023
3. Baroni Marco, Bernardi Raffaella, and Zamparelli Roberto (2014). Frege in space: A program for compositional distributional semantics. [Figura] Linguistic Issues in Language Technologies. CSLI Publications, Stanford, CA.
4. Baroni Marco, Bernardi Raffaella, and Zamparelli Roberto (2014b). Frege in space: A program for compositional distributional semantics. Linguistic Issues in Language Technologies. CSLI Publications, Stanford, CA.
5. Batchkarov Miroslav, Kober Thomas, Reffin Jeremy, Weedsand Julie and Weir David. (2016). A critique of word similarity as a method for evaluating distributional semantic models. First Workshop on Evaluating Vector Space Representations for NLP, Vol. 1, 7-12. enero 2018, De Association for Computational Linguistics Base de datos.
6. Berg-Kirkpatrick, T., Burkett, D., & Klein, D. (2012). An Empirical Investigation of Statistical Significance in NLP. EMNLP-CoNLL.
7. Bird, S. (2006). NLTK: the natural language toolkit. Proceedings of the COLING/ACL on Interactive presentation sessions (p./pp. 69--72), Stroudsburg, PA, USA: Association for Computational Linguistics.
8. Cortez Vásquez, A., Vega huerta, H., Pariona Quispe, J., & Huayna, A. (2014). Procesamiento de lenguaje natural. Revista de investigación de Sistemas e Informática, 6(2), 45 - 54. Recuperado de http://revistasinvestigacion.unmsm.edu.pe/index.php/sistem/article/view/5923/5121

9. Dacheng Tao, Xiaoou Tang, Xuelong Li, Xindong Wu. (2006). Asymmetric bagging and random subspace for support vector machines-based relevance feedback in image retrieval. IEEE Transactions on Pattern Analysis and Machine Intelligence, 28, 1088-1099. De IEEE Computer Society Base de datos.
10. Dacheng Tao, Xiaoou Tang, Xuelong Li, Xindong Wu. (2006). Asymmetric bagging and random subspace for support vector machines-based relevance feedback in image retrieval. [Figura] De IEEE Computer Society Base de datos.
11. Daniela Gerz, Ivan Vulić, Felix Hill, Roi Reichart and Anna Korhonen. (2016). SimVerb-3500: A Large-Scale Evaluation Set of Verb Similarity. Of EMNLP
12. Dongqiang Yang and David M. W. Powers (2006) Verb similarity on the taxonomy of wordnet. In 3rd International WordNet Conference
13. Duan, K., Parikh, D., Crandall, D.J., & Grauman, K. (2012). Discovering localized attributes for fine-grained recognition. 2012 IEEE Conference on Computer Vision and Pattern Recognition, 3474-3481.
14. Dubiau Luciana (2013) Procesamiento de Lenguaje Natural en
15. Duţă Oana Adriana. (2009). La semántica léxica y la oposición de sentido: un enfoque teórico. Editura Universitaria Craiova, Vol. 2, 8. enero 2018, De Facultatea de Litere - Universitatea din Craiova Base de datos.
16. Elia Bruni, Gemma Boleda, Marco Baroni, and Nam-Khanh Tran (2012). Distributional semantics in tech-nicolor. InProc. of ACL
17. Eneko Agirre, Enrique Alfonseca, Keith Hall, Jana Kravalova, Marius Pasca, Aitor Soroa, A Study on Similarity and Relatedness Using Distributional and WordNet-based Approaches, In Proceedings of NAACL-HLT 2009.
18. Escandell M. (2009). Apuntes de semántica léxica. [Figura]. España, Uned. Universidad nacional de educación a distancia.
19. Escandell Vidal María Victoria. (2007). Apuntes de semántica léxica. España: Uned. Universidad nacional de educación a distancia.
20. Faruqui Manaal and Dyer Chris. (2014). Community Evaluation and Exchange of Word Vectors at wordvectors.org. Association for Computational

Linguistics: System Demonstrations, Proceedings of the 52nd Annual Meeting, 6. enero 2018, De wordvectors.org Base de datos.

21. Faruqui Manaal, Tsvetkov Yulia, Rastogi Pushpendre and Dyer Chris. (2016). Problems with Evaluation of Word Embeddings Using Word Similarity Tasks. First Workshop on Evaluating Vector Space Representations for NLP, Vol. 1, 30-35. enero 2018, De Association for Computational Linguistics Base de datos.
22. George A. Miller & Walter G. Charles (2007) Contextual correlates of semantic similarity, Language and Cognitive Processes, 6:1, 1-28, DOI: 10.1080/01690969108406936
23. Guo, Y., Ding, G., Jin, X., & Wang, J. (2015). Learning Predictable and Discriminative Attributes for Visual Recognition. AAAI.
24. Guy Halawi, Gideon Dror, Evgeniy Gabrilovich and Yehuda Koren (2012) Large-scale learning of word relatedness with constraints. InProc. of SIGKDD
25. Halawi, G., Dror, G., Gabrilovich, E. & Koren, Y. (2012). Large-scale learning of word relatedness with constraints. In Q. Yang, D. Agarwal & J. Pei (eds.), KDD (p./pp. 1406-1414), : ACM. ISBN: 978-1-4503-1462-6
26. Hernández M.& Gómez J. (2013). Aplicaciones de Procesamiento de Lenguaje Natural. Revista Politécnica, 31, 87–96. febrero 2018, De Repositorio Institucional de la Universidad de Alicante Base de datos.
27. Kira Radinsky, Eugene Agichtein, Evgeniy Gabrilovich and Shaul Markovitch (2011) A word at a time: computing word relatedness using temporal semantic analysis. InProc. of WWW
28. Krebs Alicia y Paperno Denis. 2016. Capturing Discriminative Attributes in a Distributional Space: Task Proposal. In Proceedings of the 1st Work-shop on Evaluating Vector-Space Representations for NLP, Vol. 1, 51–54. De Association for Computational Linguistics Base de datos.
29. Lai, S., Liu, K., Xu, L., & Zhao, J. (2016). How to Generate a Good Word Embedding. IEEE Intelligent Systems, 31, 5-14.

30. Lazaridou, A., Pham, N. T., & Baroni, M. (2016). The red one!: On learning to refer to things based on their discriminative properties. arXiv preprint arXiv:1603.02618.
31. Lev Finkelstein, Evgeniy Gabrilovich, Yossi Matias, Ehud Rivlin, Zach Solan, Gadi Wolfman, and Eytan Ruppin, "Placing Search in Context: The Concept Revisited", ACM Transactions on Information Systems, 20(1):116-131
32. Lofi Christoph. (2015). Measuring Semantic Similarity and Relatedness with Distributional and Knowledge -based Approaches. Information and Media Technologies Vol. 10, 493-501. DOI: https://doi.org/10.11185/imt.10.493
33. Malagón Luque Constantino (2003). Clasificadores bayesianos. El algoritmo Naïve Bayes. Universidad Nebrija Sitio web: https://www.nebrija.es/~cmalagon/inco/Apuntes/bayesian_learning.pdf
34. Mikolov, T., Chen, K., Corrado, G. & Dean, J. (2013a). Efficient Estimation of Word Representations in Vector Space. CoRR, abs/1301.3781.
35. Mikolov, T., Yih, W.-t. & Zweig, G. (2013b). Linguistic Regularities in Continuous Space Word Representations. HLT-NAACL (p./pp. 746--751).
36. Mille r, G. A., Beckwith, R., Fellbaum, C., Gross, D. & Miller, K. (1990) Introduction to WordNet: An On-line Lexical Database. International Journal of Lexicography, 3(4):235-244.
37. Miller George A. (1995). WordNet: A Lexical Database for English. Communications of the ACM Vol. 38, No. 11: 39-41.
38. Minh-Thang Luong, Richard Socher and Christo-pher D. Manning (2013) Better word representations with recursive neural networks for morphology. InProc. of CoNLL.
39. Ng, A. & Jordan, M. (2001). On Discriminative vs. Generative Classifiers: A Comparison of Logistic Regression and Naive Bayes. In Advances in Neural Information Processing Systems (NIPS), Vol. 14.
40. Oded Avraham & Yoav Goldberg. (2017). The Interplay of Semantics and Morphology in Word Embeddings. Association for Computational Linguistics Sitio web: http://aclweb.org/anthology/E17-2067

41. Panchenko Alexander. (2017). Unsupervised Knowledge-Free Word Sense Disambiguation. [Figura]. Recuperado de https://www.slideshare.net/alexanderpanchenko/unsupervised-knowledgefree-word-sense-disambiguation
42. Pennington, J., Socher, R. & Manning, C. D. (2014). Glove: Global Vectors for Word Representation. EMNLP (p./pp. 1532--1543).
43. Perone Christian. (2013). Machine Learning: Cosine Similarity for Vector Space Models (Part III). [Figura] Recuperado de: http://blog.christianperone.com/2013/09/machine-learning-cosine-similarity-for-vector-space-models-part-iii/
44. Polanco Fernández Daniel Francisco (2000). Evaluación y mejora de un sistema automático de análisis sintagmático (Proyecto fin de carrera). Universidad Politécnica De Madrid, España.
45. Rubenstein, H., and Goodenough, J. 1965. Contextual correlates of synonymy. Communications of the ACM 8:627-633.
46. Simon Baker, Roi Reichart, and Anna Korhonen (2014) An unsupervised model for instance level subcatego-rization acquisition. InProc. of EMNLP
47. Sistemas de Análisis de Sentimientos (Tesis de Grado de Ingeniería en Informática). Facultad de ingeniería Universidad de buenos aires, Argentina
48. Trask, A., Michalak, P., & Liu, J. (2015). sense2vec - A Fast and Accurate Method for Word Sense Disambiguation In Neural Word Embeddings. CoRR, abs/1511.06388.
49. Trask, A., Michalak, P., & Liu, J. (2015). sense2vec - A Fast and Accurate Method for Word Sense Disambiguation In Neural Word Embeddings. [Figura] CoRR, abs/1511.06388.
50. Tversky, A. (1977). Features of similarity. Psychological Review, 84(4), 327-352. http://dx.doi.org/10.1037/0033-295X.84.4.327
51. Verspoor, Karin and Cohen, Kevin Bretonnel. (2013). Natural Language Processing, Encyclopedia of Systems Biology, pp. 1495-1498. DOI: http://doi.org/10.1007/978-1-4419-9863-7_158

52. Vylomova, E., Rimell, L., Cohn, T., & Baldwin, T. (2016). Take and Took, Gaggle and Goose, Book and Read: Evaluating the Utility of Vector Differences for Lexical Relation Learning. CoRR, abs/1509.01692.
53. Wang, Y., & Mori, G. (2010, September). A discriminative latent model of object classes and attributes. In European Conference on Computer Vision (pp. 155-168). Springer, Berlin, Heidelberg.
54. Warin, M. (2004). Using WordNet and Semantic Similarity to Disambiguate an Ontology.
55. Y. Ng Andrew and I. Jordan Michael. (2008). On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes. Neural Processing Letters, 7, DOI: 10.1007/s11063-008-9088-7.