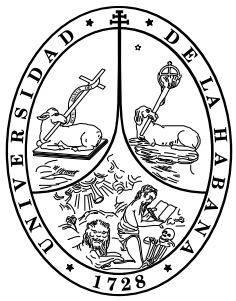
Universidad de La Habana
Facultad de Matemática y Computación



Aprendizaje de representaciones semánticas a partir de bases del conocimiento

Autor: **Aryan Curiel Pardo**

Tutores: **MSc. Alejandro Piad Morffis**
Lic. Juan Pablo Consuegra Ayala

Trabajo de Diploma
presentado en opción al título de
Licenciado en Ciencia de la Computación

Junio de 2018

"The Road goes ever on and on
Down from the door where it began.
Now far ahead the Road has gone,
And I must follow, if I can."

J. R. R. Tolkien

Agradecimientos

No existe una sola persona en la tierra totalmente independiente. Desde el antes del tiempo el ser humano vivió y creció en grupos. Yo no soy la excepción. Esta etapa de mi vida ha sido agotadora, llena de preocupaciones, sustos y respiros, pero tan divertida! Y lo mejor ha sido esa dependencia. Esa que sin pensarlo ni pedirlo me ha permitido vivir – no sobrevivir – cada uno de esos instantes oscilantes entre ruidos y sorpresas. A ellos quiero dedicarles esta tesis.

A mi mamá y a mi papá, que con abrazos, besos, sonrisas y turismo de vida me ayudaron a ser quien soy. Hacerlos sentir orgullosos es y siempre será una de las principales constantes de la ecuación que juntos me ayudan a vivir.

A Patry, mi hermanita preferida. Ser tu hermano es uno de esos momentos con ruidos y sorpresas. Que la vida siempre nos mantenga juntos. A Poti, que no puede leer esto. Recogerte fue una de las mejores decisiones que tomé en mi vida. Gracias por tu cariño.

Al resto de mi familia, que cada vez que me veían comenzaban siempre con la misma pregunta: "¿... y cómo te va con la tesis?", lo que llevaba igual la misma respuesta: "ahí..."

A Magalys, Rafa y mi abuela Xiomara. Gracias a ustedes pude trabajar en la tesis desde cualquier parte. Gracias por su sacrificio y confianza.

A todos los que vivieron esta etapa de mi vida junto a mí en la facultad, sobre todo estos últimos momentos de tensión y chismes. En especial a Amelia, Rocío, Eddy, Dainel y todos aquellos que incluso con una pequeña charla me ayudaron a sentir que aún tenía tiempo. A Jean por la constante preocupación e interés, y por brindarnos más tiempo cuando hacía falta. A JJ por los empujones y consejos en los momentos oportunos, y por su constante disposición a cada pregunta, miedo y curiosidad que tuve en mis 5 años de la carrera. A los profes Idania, Katrib, Lester, Somoza, Ludwig, Alberto, Yudivian, Fernan, Suilan, Marcelo, y cada uno de esos profesores que con su experiencia me permitieron crecer.

A Piad. Además del mejor tutor fuiste el diazepam que calmó cada uno de los momentos de tensión durante esta tesis. Gracias por hacerla sentir más placentera. Perdóname por no haber sido más responsable. El respeto y la estima que te tengo fue una de las principales razones por las que cuando hizo falta me puse las pilas.

A Tania, la mejor suegra del mundo. Gracias por la locura que aportaste a cada uno de esos momentos de tensión, y por toda la ayuda que me brindaste todos los días que jodí en tu casa.

A cada uno de los integrantes de mi manada, a Victor por los momentos de "relax", a Manu por sus regaños y apoyo. A Gaby por ese cariño innegable que durante años me ha hecho sentir afortunado de haber ido a *cnx*. A Fabio, mi negro, gracias por esos intensivos. Que muchos países tengan el placer de conocernos juntos. No sé que habría hecho sin ellos. A Roxy por su lucha constante y su amistad. A Alfre por sus ayuda en esos momentos en que yo debía ayudarlo. Lástima no graduarnos juntos, pero espero que algún día trabajemos de verdad uno al lado del otro. A Isa. Eres especial y lo sabes. Por todos los momentos que nunca vivimos pero que seguro viviremos algún día. Espero nunca convertirme en un "tumor" para ti. Un abrazo bien grande a todos.

Y por último, a mi bobita, a Tali. Sin ti no sé quién sería ahora mismo, ni quiero saberlo. Esta es la parte que más me ha costado escribir, pues nada de lo que teclee será suficiente. Gracias por tener la paciencia para escuchar mis quejidos y toda esa palabrería de la tesis que intentabas cada día entender. Gracias por todas las veces que cocinaste solita. Disculpa cada una de las noches que no pudimos acostarnos abrazados para poder adelantar. Gracias por hacer de mi cumple uno de los más divertidos a pesar de esta tesis y de las lluvias. Nada, que contigo a mi lado no hay proyecto ni tempestad que me evite ser feliz junto a ti. Espero que alguna vez les leamos a nuestros nietos estas palabras todo arrugaditos y con canas. Te quiero y te amo mucho, bobita.

Opinión del tutor

La representación de conocimiento es una de las áreas pilares en la inteligencia artificial, desde sus orígenes. Actualmente es reconocido por la comunidad, que una buena representación de los datos puede tener una influencia significativa, quizá incluso decisiva, en el rendimiento de cualquier proceso de aprendizaje. A grandes razgos se pueden distinguir los enfoques basados en conocimiento experto, que explotan características del dominio, y los enfoques estadísticos que explotan patrones recurrentes en los datos. Con el crecimiento del volumen de datos y el poder de cómputo, y la llegada de la era del aprendizaje profundo, han comenzado a aparecer enfoques que intentan combinar lo mejor de ambos mundos. Una de las técnicas más populares en este sentido, son los llamados embeddings de entidades, que permiten representar objetos con una semántica compleja mediante un vector numérico que captura propiedades relacionadas con el contexto donde dicho objeto aparece en cierta red semántica.

En esta problemática se sitúa la tesis de Aryan, presentando una propuesta novedosa para aprender representaciones vectoriales a partir de una base de conocimientos arbitraria. Aunque la tesis de Aryan se enfoca particularmente en la red semántica WordNet, las técnicas propuestas son fácilmente adaptables a otras bases de conocimiento con estructuras disímiles. El principal aporte de esta tesis, en contraste con otros modelos de embedding de entidades existentes, es la forma de representar relaciones. Los resultados obtenidos demuestran que las representaciones aprendidas capturan propiedas semánticas de las relaciones, tales como la simetría, la reciprocidad, y la transitividad. Aunque iniciales, estos resultados abren las puertas a estudios más profundos que permitan utilizar estas representaciones en contextos donde el conocimiento del dominio tenga un peso importante, y la cantidad de datos existentes sea poca, justamente el tipo de situaciones donde los embeddings clásicos pierden relevancia.

Durante el desarrollo de la tesis Aryan demostró independencia y creatividad para lidiar con los problemas encontrados. Tuvo que dominar concep-

tos y tecnologías del estado del arte, y conoció las complejidades de trabajar con grandes volúmenes de datos y modelos de aprendizaje costosos sin contar con toda la infraestructura tecnológica necesaria. Estas realidades, más allá de los problemas teóricos que hubo de resolver, le sirven también de aprendizaje para su desarrollo futuro. El documento escrito muestra también la capacidad adquirida para presentar resultados de investigación de forma concisa y coherente.

Aryan ha sido alumno ayudante de la asignatura de Programación desde los primeros años de la carrera. En esos años he podido comprobar su interés y dedicación por temas muy diversos. Este último ejercicio demuestra que ya ha adquirido la madurez necesaria para desarrollar proyectos de alta complejidad con calidad y esmero. Como tutor, estoy muy complacido por los resultados obtenidos, y por el trabajo realizado con Aryan, que aunque no estuvo excento de obstáculos, logró superar con creces los objetivos propuestos. Por estos motivos estoy convencido de que Aryan será un excelente profesional de la Ciencia de la Computación.

MSc. Alejandro Piad Morffis
Facultad de Matemática y Computación
Universidad de La Habana

Resumen

El Procesamiento de Lenguaje Natural usando aprendizaje de máquinas es un área de trabajo muy investigada en la actualidad en la Inteligencia Artificial. La representación de los textos como vectores reales para su uso en los distintos algoritmos es de suma importancia, pues del método usado depende cuánta información semántica del texto se puede representar en los vectores resultantes. En el siguiente trabajo se realiza una investigación sobre el estado del arte actual de los principales métodos de representación de textos. Se analizan los beneficios del uso de una base del conocimiento estructurada en lugar de grandes corpus de texto plano. Se presenta un modelo de aprendizaje basado en *Word Embedings* y *Entity Embeddings* capaz de representar la información estructurada de ontologías como WordNet y de aprender las propiedades y características presentes en sus relaciones.

Abstract

Natural Language Processing using machine learning is a work area that is highly researched in Artificial Intelligence. The representation of the texts as real vectors for use in the different algorithms is very important, since the method used depends on how much semantic information of the text can be represented in the resulting vectors. In the following work an investigation is made on the state of the art of the main methods of representation of texts. We analyze the benefits of using a structured knowledge base instead of large corpus of plain text. A learning model based on Word Embedings and Entity Embeddings is presented, able to represent the structured information of ontologies such as WordNet and to learn the properties and characteristics present in their relationships.

Índice general

Índice de figuras

Índice de tablas

Introducción

La principal fuente de información en la actualidad es Internet. El procesamiento de todo ese conocimiento puede brindar grandes resultados para casi cualquier propósito. La mayor parte de esa información se encuentra en forma de texto. Existen muchos problemas en el procesamiento de textos como la minería de opiniones, la detección de tendencias, tópicos y sentimientos, generación de textos, entre otros. Dentro de la Inteligencia Artificial, el Procesamiento del Lenguaje Natural es una de las áreas de estudio que más ha avanzado en las soluciones a esos problemas con el uso de algoritmos de Machine Learning y Deep Learning. Estas últimas están conformadas por dos momentos de decisión: el algoritmo a usar y el método para representar los datos. Este último se vuelve de gran complejidad cuando los datos son textos.

En el Grupo de Inteligencia Artificial de la Universidad de La Habana existe una línea de investigación activa en el Procesamiento del Lenguaje Natural. En este marco se han llevado a cabo varias investigaciones en el área de detección de idiomas [2], trabajo con tópicos [1], análisis de sentimientos y opiniones [11, 12], entidades nombradas [14], entre otros. Durante el desarrollo de estas se ha podido confirmar la inefectividad de muchos de los métodos de representación de textos para los algoritmos de Machine Learning. Otra línea de investigación [30] ha sido el uso de las ontologías, principalmente WordNet, donde se han podido demostrar los beneficios del aprovechamiento de las estructuras y relaciones semánticas presentes en ella. En la actualidad se está comenzando una nueva línea de investigación en Deep Learning que tiene como objetivo principal investigar

Problemática

La representación de textos para su uso en algoritmos de Machine Learning es un problema extremadamente complejo por la dificultad de expresar

la semántica en términos computacionales. Uno de los métodos más simples y populares de hacerlo es Bolsa de Palabras, pero las altas dimensiones y la poca representación semántica que resulta de su uso impiden lograr mejores resultados. Existen otros enfoques, pero se hace difícil lograr un balance entre las dimensiones de la representación resultante y la completitud de esta de acuerdo a la información semántica que aporta el texto. Esto trae como consecuencia lentitud en la ejecución de los algoritmos y poca efectividad en los algoritmos de aprendizaje usados.

Una de las técnicas más novedosas en la actualidad son los Word Embeddings, capaces de lograr representaciones de mucha menor dimensión que los enfoques anteriores y sin la necesidad de usar un corpus. Sus desventajas están en su necesidad por grandes cantidades de datos, además de la ausencia de una noción explícita de las relaciones semánticas presente en los textos.

Hipótesis

Acorde a las líneas actuales de investigación del Grupo de Inteligencia Artificial y conociendo las ventajas y desventajas los enfoques actuales, surge la idea de sumar la eficiencia de los Word Embeddings y la información semántica de las ontologías como WordNet, lo que parece prometer una mejora en la efectividad de los algoritmos de aprendizaje en el Procesamiento del Lenguaje Natural.

Objetivos

Diseñar un método de representación de textos para algoritmos de aprendizaje usando Word Embeddings para la codificación y WordNet como base semántica.

Objetivos Específicos

1. Realizar un estudio de la literatura especializada y obtener una actualización del estado del arte en la representación de textos para algoritmos de Machine Learning.

2. Analizar la factibilidad y las formas de unir Word Embedding y WordNet.

3. Diseñar un modelo de aprendizaje que brinde solución al problema planteado.

4. Implementar un prototipo computacional usando Keras y Tensorflow.

5. Analizar mediante la experimentación las características y beneficios de el enfoque tomado para el desarrollo del modelo.

Estructura de la tesis

En el Capítulo 1 se realiza una investigación del estado del arte en el campo de la representación de textos, haciendo un recorrido más profundo en los principales enfoques de los estudios realizados en esta rama del Procesamiento de Lenguaje Natural. En el Capítulo 2 se describe la propuesta de la presente investigación que consiste en el desarrollo de un modelo de aprendizaje que use los beneficios de una base del conocimiento bien estructurada. En el Capítulo 3 se explican los resultados de los experimentos realizados. Estos consisten en distintas ejecuciones del modelo en casos específicos con el objetivo de analizar las propiedades presentes en los *embeddings* resultantes. Finalmente se presentan las conclusiones de la investigación y algunas recomendaciones para futuros trabajos.

Capítulo 1

Representación de Textos

Las computadores son mucho mejores en ciertas tareas que los humanos, como en buscar entre grandes cantidad de datos o realizar complejos cálculos. Sin embargo todavía los investigadores no han logrado que estas sean tan capaces como nosotros en el entendimiento del significado de las palabras debido a la enorme dependencia de este problema con el contexto en que se encuentra. Replicar todo lo representado en la semántica del lenguaje natural continúa siendo un problema sin resolver en las ciencas de la computación.

La mayoría de los enfoques consideran la llamada *Teoría distribucional de la semántica* como base para sus investigaciones. Esta hipotetiza que las palabras que están distribuidas de la misma manera en el texto tendrán significados similares, por lo que las palabras *teclado* y *monitor* deben ocurrir juntas en el texto con frecuencia, pues se refieren a objetos relacionados en el mundo real. Este enfoque semántico defiende la idea del lingüista JR Firth: "*a word is characterized by the company it keeps*" [13].

1.1. Representación Distribucional

Los primeros enfoques en surgir fueron los de representación distribucional. Estos usan una matriz de coocurrencia F de tamaño $W \times C$ donde W es el tamaño del vocabulario y C la cantidad de contextos considerados. Cada fila F_w es la representación inicial de la palabra w, y cada columna F_c es la representación de un contexto c. Algunos trabajos [34, 39] describen un grupo de posibles diseños a tener en cuenta para la construcción de la matriz F. Esto se puede hacer dependiendo del tipo de contexto, donde podemos encontrar las siguientes opciones:

1. ***left window***: El contexto se escoge como las k palabras más cercanas

que ocurren antes de la palabra actual.

2. ***right window***: El contexto se escoge como las k palabras más cercanas que ocurren después de la palabra actual.

3. ***size of window***:El contexto se escoge como las k palabras más cercanas que ocurren antes y después de la palabra actual.

y considerando el modo en que se cuenta la frecuencia de los términos en un contexto dado, donde se encuentran:

1. ***raw***: se toma el conteo exacto de las apariciones de una palabra en un contexto dado

2. ***binary***: se considera solo si aparece o no una palabra en un contexto dado

3. ***tf - idf***: se usan los valores *tf* e *idf* de los términos

Los Mapas Semánticos Auto-organizados (*Self-organizing semantic maps*) [33] son otra de las técnicas de representación distribucional. En este se convierten las palabras a vectores bidimensionales logrando que palabras relacionadas sintáctica y semánticamente se encuentren cercanas en dicha representación [16, 15].

También se pueden encontrar trabajos como LSA (*Latent Semantic Analysis*) [10, 19], LSI y LDA (*Latent Dirichlet Allocation*) [5], que presentan una representación distribucional en la cual cada columna de la matriz F representa el contexto de un documento, contrario a los enfoques anteriores donde cada columna representaba el contexto de una palabra.

Otro enfoque distribucional es el Hiper-espacio Análogo al Lenguaje (*Hyperspace Analogue to Language*) [23, 22]. En este se calcula la matriz F a partir de un corpus de 160 millones de palabras con un vocabulario de tamaño W con 70 mil tipos de palabras. Para esto se tienen en cuenta las 10 palabras anteriores y posteriores a cada palabra, quedándose luego con las columnas con mayor varianza.

El principal problema de todos estos enfoques es el costo en memoria de guardar la matriz F de coocurrencia. Una opción podría ser transformarla en una segunda matriz para reducir su tamaño, pero este puede resultar extremadamente ineficiente. Hay que tener en cuenta que la mayoría de estos métodos distribucionales no tienen en cuenta las relaciones semánticas entre las palabras, contrario a otros enfoques que pueden resultar en mayor información semántica.

1.2. Representación de textos basada en *Clustering*

Otra forma de representar textos es realizando clustering sobre las palabras. Estos métodos se pueden mezclar con los distribucionales. Por ejemplo, construyendo una matriz de coocurrencia y luego la transformándola en clustering [29].

El algoritmo de Brown [6] es un algoritmo de clustering jerárquico que agrupa palabras maximizando la información mutua de los bigramas. Este ha sido usado satisfactoriamente en muchas aplicaciones entre las que podemos encontrar el Reconocimiento de Entidades Nombradas (*Named Entity Recognition*) [21, 31], *Dependency Parsing* [18, 37] y *Semantic Dependency Parsing* [42]. Otros trabajos sobre el algoritmo de Brown lo han usado sobre trigamas [24] mientras otros aprenden clusterings jerárquicos a partir de frases [40].

Uno de los beneficios del algoritmo de Brown es su naturaleza jerárquica, lo que permite escoger las clases de las palabras en múltiples niveles de la jerarquía. Por otro lado, una de su desventajas es que está solo basado en la información que ofrecen los bigramas y no tiene en cuenta el uso de las palabras en un contexto más general.

1.3. Representación distribuida

Otro enfoque es el de aprender una representación distribuida. Estos métodos son conocidos como *Word Embeddings*. Contrario a las representaciones distribucionales, estos son densos y de bajas dimensiones. Cada dimensión representa una característica latente de la palabra. Son típicamente construidos usando redes neuronales [3].

Los trabajos sobre este enfoque han propuesto eliminar la dependencia linear al tamaño del vocabulario y permitir el uso de grandes corpus de entrenamiento. Este resultado muestra un gran paso de avance teniendo en cuenta que históricamente el entrenamiento y el *testing* en redes neuronales ha sido exremadamente lento, comparable con el tamaño del vocabulario por modelo a computar [4].

Una propuesta diferente fue la de Mnih y Hinton [26], donde presentan una función de pérdida log-bilineal. Dicho modelo concatena los embeddings de las primeras $n-1$ palabras de un n-grama dado y aprende un modelo lineal para predecir el embedding de la última palabra. Luego, en otra publicación [27] propusieron el modelo HLBL (*Hierarchical log-bilinear*), en

el cual se utiliza un enfoque jerárquico similar a otros trabajos [28] con el objetivo de acelerar la evaluación del modelo durante el entrenamiento y la fase de *testing*. Esta propuesta poda el espacio de búsqueda de la próxima palabra dvidiendo la predicción en un grupo de predicciones que filtran las regiones del espacio. Otro de los embeddings interesantes en esta área es el embedding de SENNA [9] el cual es generado usando un modelo discrimador y no probabilístico. Esto lo hace leyendo un n-grama del corpus en cada actualización del entrenamiento y concatenando los embeddings aprendidos de las n palabras. Luego se crea un ruido usando un n-grama *corrupto* que es construido reemplazando la palabra en el medio de este por una escogida aleatoriamente del vocabulario. Entonces se logra que el modelo aprenda una función que diferencie entre ambas frases.

Una propuesta basada en la anterior [38] realizaba lo mismo que el *embedding* de SENNA con algunas diferencias. La principal es que corrompían la útlima palabra de cada n-grama en vez de la palabra del medio. Además, notaron que el uso de *embeddings* aprovechando las características típicas de los problemas de NLP parecía mejorar el rendimiento en problemas como los de *Named Entity Recognition*.

1.3.1. Skip-gram y Word2Vec

Skip-gram [25] es una de las arquitecturas de modelos más efectivas para el aprendizaje de representaciones distribuidas de las palabras. Este nuevo enfoque intenta mimizar la complejidad computacional de este proceso. En vez de intentar predecir la palabra actual basándose en un contexto, Skip-gram intenta maximizar la clasificación de una palabra basándose en otra palabra de la misma oración. Dicho de otra manera, usa cada palabra como entrada de un clasificador log-lineal con una capa de proyección contínua, y predice palabras que se encuentren en un rango antes y después de la palabra en cuestión.

Los estudios de Mikolov con Skip-gram encontraron que incrementar el rango en el que se predicen las palabras a partir de la actual aumenta la efectivdad de los resutados, pero también incrementa la complejidad computacional del modelo. Además, dado que usualmente las palabras más distantes en los textos están menos relacionadas a la palabra actual que las más cercanas, en Skip-gram se le da menos peso a las palabras más alejadas haciendo que la probabilidad de escoger una palabra determinada para el entrenamiento sea dependiente de su cercanía a la palabra actual.

El uso de esta arquitectura permite usar simples operaciones algebraicas para resonder preguntas como: ¿cuál es la palabra más similar a *small* en el

mismo sentido en que *biggest* lo es con *big*? De este modo podemos computar el vector:

$$X = vector(\text{"}biggest\text{"}) - vector(\text{"}big\text{"}) + vector(\text{"}small\text{"}) \qquad (1.1)$$

si buscamos en el espacio por el vector más cercano a X teniendo en cuenta la distancia coseno y los vectores de las palabras fueron bien entrenados, podemos notar que el vector encontrado es efectivamente el que representa a la palabra *smallest*. Usando la arquitectura de Skip-Gram, su autor Tomas Mikolov y otros integrantes de Google entrenaron su modelo usando un corpus de Google News que contenía cerca de 6 billones de tokens restringiendo el vocabulario a 1 millón de las palabras más frecuentes. Este proyecto recibió el nombre de Word2Vec y mostró resultados favorables, pero 1 pesar de estos, aun no son capaces de etiquetar las relaciones semánticas entre las palabras.

1.3.2. Entity Embeddings

Es innegable el avance que ha ocurrido en esta área del Procesamiento del Lenguaje Natural; sin embargo el trabajo existente aun es limitado en el uso de conocimiento estructurado con el objetivo de lograr mejores representaciones. Por ejemplo, los embeddings de palabras y frases son popularmente inducidos a partir de corpus grandes de textos planos. Además, en la mayoría se asume que cada palabra tiene solo un significado a pesar del contexto, por lo que es respresentada por un solo vector en un espacio semántico, por ejemplo *charlotte (ciudad) vs. charlotte (nombre).*

Otra dificultad es que el espacio usualmente solo contiene vectores de palabras simples. Los vectores de expresiones multipalabras (MWEs) son normalmente obtenidos a partir de la media de los vectores de las palabras simples que forman las expresiones. Esto usualmente resulta en representaciones inexactas, sobre todo si el significado del MWE es diferente a la composición de los significados de las palabras individuales que lo componen. Por ejemplo $vector(northcarolina) vs. vector(north) + vector(carolina)$. Además, palabras que son usadas para referirse a un mismo concepto tendrán embeddings diferentes, como *america* y *usa*.

Muchos trabajos se han enfocado en el aprendizaje distribuido de conceptos y entidades con el objetivo de eliminar las limitaciones mencionadas anteriormente. Utilizan mayormente bases del conocimiento basadas en texto plano como la Wikipedia. En general, estos métodos actuales pueden dividirse en dos categorías: aquellos que aprenden solo conceptos de la base del conocimiento [17, 20, 32], y los que aprenden conjuntamente palabras y

conceptos dentro del mismo espacio semántico [8, 41, 7, 35]. Algunos trabajos modifican el *Skip-gram* de Word2Vec para lograr el aprendizaje de entidades [36].

Muchos trabajos han seguidos estas ideas para mejorar el comprotamienteo de los algoritmos de apendizaje en el procesamiento de lenguaje natural. Uno de ellos es el Entity Embedding Jerárquico [17] en el que construyen su arquitectura a partir del *skip-gram* mencionado anteriormente, pero generalizan la definición del contexto y la medida de similaridad para Entity Embeddings Jerárquicos. Como en una base del conocimiento de artículos enciclopédicos normalmente se enfonca en describir una sola entidad a partir de cada artículo, ellos extienden el contexto de la entidad al artículo entero y obtinen un conjunto de pares de entidades formados por la entidad principal y una entidad decundaria que ocurre en el contexto de la entidad principal. Luego usan dicho conjunto para aprender un vector para las entidades principales y otro para las entidades de su contexto. Este proyecto mostró buenos resultados en los problemas de Enlace de Entidades (Entity Linking) y en la Búsqueda de Entidades (Entity Search).

Otro trabajo [41] propone un método de embedding diseñado específicamente para el problema de Desambiguación de Entidades Nombradas (*Named Entity Disambiguation - NED*) mapeando palabras y entidades juntas en el mismo espacio vectorial. Para esto extienden el *skip-gram* usando dos modelos: el *KB graph model* que aprende relaciones entre entidades a partir de las estructuras de enlace de la base del conocimiento usada, y el *anchor context model* que busca alinear los vectores de modo que las palabras y entidades similares aperecen cercanas unas a las otras en el mismo espacio vectorial.

Como se pudo ver, los *Entity Embeddings* han mostrado buenos resultado en los trabajos que intentan resolver problemas de reconocimiento de entidades. Por otro lado, a pesar del avance en la representación de textos, no se ha logrado nombrar y organizar la estructura semántica contenida en los textos. Teniendo en cuenta lo anterior, desarrollar un modelo de aprendizaje basado en *Entity Embeddings* para el reconocimiento de relaciones y estructuras semánticas, puede mostrar características y propiedades útiles en la investigación y solución de múltiples problemas en el procesamiento de lenguaje natural.

Capítulo 2

Propuesta de solución

A continuación se describirán los componentes teóricos de la propuesta de modelo de representación distribucional que se presenta en esta investigación. Esta propuesta usa los beneficios mostrados en los trabajos de *Entity Embedding* usando las estructura presente en una ontología como WordNet. Primeramente se dará a conocer la estructura y las propiedades de dicha ontología, así como de las principales relaciones que podemos encontrar entre sus elementos. Luego se explicará el proceso de extracción de la información que brinda, y de la construcción de un corpus a partir de ella que se usará para entrenar y probar el modelo propuesto. Por último se expondrán la estructura de dicho odelo y sus beneficios.

2.1. WordNet y sus relaciones

WordNet fue creado en la Universidad de Princeton en 1986. Desde entonces ha formado parte del procesamiento del lenguaje natural. Es un grafo de términos con su propia lógica y jerga. Su estructura gira alrededor del concepto [1] de "conjunto de sinónimos" (o *synsets*), el cual es una colección no ordenada de "palabras y frases cognitivamente sinónimas". Los elementos que componen los *Synsets* son llamados *Lemmas*, los cuales representan los diferentes significados.

Otra de las características de WordNet es su sistema de relaciones. Estas pueden ser entre *Synsets* o entre *Lemmas*. Las relaciones que ocurren entre *Lemmas* reciben el nombre de *reaciones léxicas*, mientras que las que ocurren entre *Synsets* son llamadas *relaciones semánticas*. La mayoría de las relaciones de WordNet conectan palabras de la misma parte de la oración, llamadas usualmente PoS (*Part of Speech*), por lo que al final consiste en

cuatro sub-redes, cada una para cada PoS: **sustantivos**, **adjetivos**, **verbos** y **adverbios**, con las relaciones que ocurren entre ellas. Algunas de estas relaciones son:

- **Synonomy**: La principal relación de WordNet es la sinonimia. Es una relación léxica que enlaza palabras con significados similares, como *roof* y *ceiling*. Un conjunto de sinónimos forma un *Synset* de WordNet. Según la definición, dos expresiones son sinónimas en un contexto lingüístico C si la sustitución de una por la otra en una expresión de C no altera su correctitud. También podemos asumir que dicha relación es simétrica y transitiva.
- **Antonym**: Dos palabras tal que sus significados son totalmente contrarios son llamados antónimos. Por ejemplo, *old* y *young*. Es importante aclarar que la antonimia es una relación léxica, por lo que relaciona formas de palabras y no una relación semántica entre significados de palabras. Un ejemplo de esto son los significados de *rise*/*ascend* y de *fall*/*descend*. Estos pueden ser conceptualmente opuestos pero no son antónimos. *Rise* y *fall* si lo son, al igual que *ascend* y *descend*, sin embargo *rise*/*descend* y *fall*/*ascend* no son aceptados como antónimos.
- **Hypernomy**: Una palabra que es más general que otra palabra, es llamada su hiperónimo. Por ejemplo, *plant* es hiperónimo de *tree*. Esta relación es transitiva, asimétrica y forma parte del conjunto de relaciones semánticas.
- **Hyponomy**: Una palabra que es más específica que otra palabra es llamada su hipónimo. Por ejemplo, *tree* es hipónimo de *plant*. Esta relación es transitiva, asimétrica y forma parte del conjunto de relaciones semánticas.
- **Meronomy**: Una palabra que es parte (miembro) del significado de otra palabra es llamada su merónimo. Por ejemplo, *bed* es merónimo de *house*.
- **Holonomy**: Una palabra que contiene a otra palabra siendo esta última merónimo de la primera, es llamada su holónimo. Por ejemplo, *Japan* es un holónimo de *Fuji* pues el monte Fuji está en Japón. Esta relación es transitiva, asimétrica y forma parte del conjunto de relaciones semánticas.
- **Entailment**: Es una relación unilateral que establece la dependencia de un verbo a otro. Por ejemplo, *snore* y *sleep*, pues la primera acción

no puede ocurrir al menos que la segunda esté ocurriendo. Esta relación es transitiva, asimétrica y forma parte del conjunto de relaciones semánticas.

- **Causal**: Esta relación establece una causalidad entre dos conceptos de verbos. Por ejemplo, *give* y *have*. La acción del primero implica la ocurrencia del segundo. Es importante notar que existe una causalidad pero no una dependencia a diferencia del *Entailment*. Esta relación es transitiva, asimétrica y forma parte del conjunto de relaciones semánticas.

Existen otras relaciones en WordNet, pero debido a su poca relevancia cuantitativa y con el objetivo de mantener simple el modelo, se decidió no considerarlas.

2.2. Construcción de la Base del Conocimiento

Como se explicó anteriormente, WordNet está estructurado como un grafo donde los nodos son los Synsets y Lemmas, y las aristas representan los distintos tipos de relaciones que pueden haber entre ellos. Dada esta estructura, se necesita representar la información que brinda dicha ontología en una colección de elementos que podamos usar en el aprendizaje de nuestro modelo.

2.2.1. Extracción desde WordNet

Para la extracción de la información de WordNet se decidió construir a partir de sus nodos y relaciones, todas las tripletas de la forma (w_1, r, w_2), donde w_1 y w_2 son palabras contenidas en WordNet (*Synsets* o *Lemmas*), y r es una relación existente entre ellos de las anteriormente mencionadas. Por ejemplo, las tripletas siguientes son posibles elementos resultantes:

- *wear* **syn** *clothing*
- *frame* **hyper** *framework*
- *atmosphere* **mero** *exosphere*

Estas tripletas se pueden construir a partir de la lista de todos los *Synsets* de WordNet. De cada uno de ellos se pueden conocer todos los *Synsets* con los cuales mantiene alguna relación semántica. También se puede conocer el listado de *Lemmas* contenidos en un *Synset* dado, los cuales serán

sinónimos entre sí. Además, de cada *Lemma* es posible conocer su listado de antónimos. Con estas operaciones ya se pueden obtener todas las parejas de palabras enlazadas en WordNet mediate alguna de las relaciones de interés mencionadas en esta investigación.

2.2.2. Generación de ruido

Además de las tripletas construidas a partir de WordNet, es necesario generar un conjunto de tripletas incorrectas. Se debe construir dicho ruido manteniendo los datos consistentes y proporcionales. Para esto, al generar cada tripleta del ruido, seleccionaremos las palabras aleatoriamente, pero correspondiéndose con su ocurrencia en los datos originales. La relación entre estas será escogida también proporcionalmente teniendo en cuenta las apariciones de dichas palabra con cada relacion en los datos. Para esto se debe calcular con anterioridad en cuantas tripletas participa cada palabra, y para cada palabra en cuantas tripletas aparece por cada relación considerada durante la extracción de la información. Siguiendo estos pasos se puede garantizar que las tripletas falsas construidas poseen la misma distribución que las originales.

2.2.3. Construcción final del corpus

Luego, a partir de ambos conjuntos de tripletas, el original y el de ruido, se construye el corpus. Lo primero que que se hace es indexar los elementos. Para esto se enumeran todas las palabras usadas en las tripletas con un identificador entero único dentro del intervalo $[0, ||W||]$, donde W es el conjunto de palabras extraídas. También se numeran las relaciones pero en el intervalo $[0, ||R||]$ donde R es el conjunto de relaciones. Además se agrega a las tripletas un valor entero v en el intervalo $[0, 1]$. De esta manera, la tripleta (w_1, r, w_2) se convierte en una 4-tupla de la forma (w_i1, r_i, w_i2) donde w_i1, r_i y w_i2 son los identificadores enteros de w_1, r y w_2 respectivamente, y v toma valor 0 si la tripleta es del conjunto generado de ruido, o toma valor 1 si es una de las tripletas originales construídas a partir de WordNet.

2.2.4. Modelo propuesto

El modelo que se propone consiste en una red neuronal con una estructura basada en las tripletas (w_1, r, w_2) mencionadas anteriormente y su correctitud marcada por el valor v. Como se puede ver en la Fig 2.1, el modelo contiene tres capas de entrada. Dos de estas manejan las entradas de las palabras w_1 y w_2, las cuales son pasadas por una capa de embedding

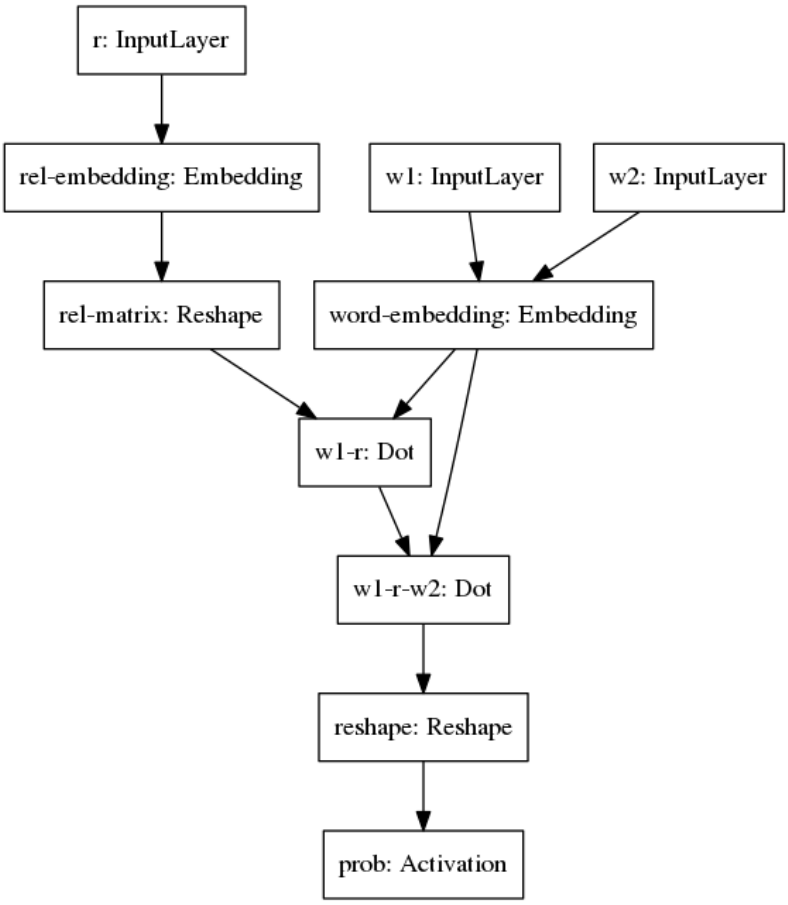


Figura 2.1: Modelo propuesto

(*word-embedding*) que se encargará de aprender las representaciones de las distintas palabras del corpus, y de enviar a las siguientes capas los vectores de tamaño K correspondientes a las dos palabras pertenecientes a la tripleta actual. De igual manera, la relación es recibida por la capa de entrada restante y pasada por una capa de embedding (*rel-embedding*) que aprenderá la representación correspondiente a la relación de la tripleta actual en un vector de tamaño K^2. Luego este último vector será pasado por una capa de *Reshape* (*rel-matrix*) que convertirá el vector de tamaño K^2 recibido a una matriz cuadrada R de tamaño $K \times K$.

Llegados a este punto del modelo se tienen dos vectores representando a las palabras w_1 y w_2, y una matriz cuadrada para la relación r, por lo que se necesita unir la información brindada por los tres elementos. Esto se puede lograr multiplicándo dichos elementos de la siguiente manera:

$$v' = vector(w_1) \times R \times vector(w_2) \tag{2.1}$$

Esta operación se puede realizar pasando la matriz R y el vector asociado a la palabra w_1 resultante del embedding por una capa *Dot* (*w1-r*). Luego el resultado de esta se puede pasar por otra capa *Dot* (*w1-r-w2*) junto con el vector de la palabra w_2, lo que trae como resultado un vector v' de una sola componente. El modelo termina quedandose con la componente única del vector resultante, para luego pasar dicho valor por una última capa de activación (*prob*) sigmoidal. El elemento resultante será comparado con el valor v asociado a la tripleta actual, logrando durante el aprendizaje con el corpus que nuestro modelo reconozca la veracidad o falsedad de las tripletas.

Capítulo 3

Experimentación

En este capítulo se muestran los experimentos realizados con el objetivo de evaluar la propuesta de modelo definida en el capítulo anterior, y los resultados de estos. Se descibe el ambiente en el que se desarrollaron, datos estadísticos del corpus construido y del ruido generado, así como algunas propiedades encontradas durante la experimentación.

3.1. Ambiente de experimentación

El modelo fue programado en el lenguaje de programación *Python*. Para el desarrollo del modelo usamos el módulo *Keras* apoyándose sobre *Tensorfow* como tecnología de *backend* para ser ejecutado. Otros módulos de Python usados fueron *matplotlib* y *numpy*

Para el trabajo con WordNet se usó el corpus con el mismo nombre brindado por el módulo *NLTK* de Python. El modelo fue entrenado y probado usando las herramientas de *Jupyter Lab* sobre un dispositivo **Intel Core i3-5005U** con 4GB de memoria RAM.

3.2. Construcción del corpus

La extracción de las tripletas (w_1, r, w_2) de WordNet descritas en el Capítulo 2, teniendo en cuenta solo las relaciones mencionadas en dicho capítulo, resulta en **528,741** elementos. La distribución de las tripletas de acuerdo a cada relación muestra un predonminio de *synononym* (Figura 3.1), mientras que las relaciones de *antonym*, *entailment* y *causal* carecen de presencia en el corpus. Sobre todo en estas dos últimas, lo cual ocurre debido al predominio de los adjetivos y sustantivos sobre los verbos, entre los caules ocurre

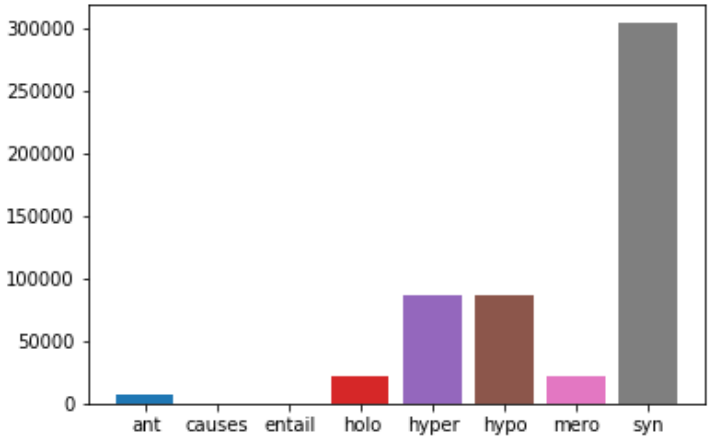


Figura 3.1: Distribución de tripletas por relación

dichas relaciones. También se puede notar el caracter simétrico existente entre las relaciones *hypernomy/hiponomy* y *meronomy/holonomy* en el emparejamiento que hay entre sus cantidades.

Si se añade aproximadamente la misma cantidad de ruido que de tripletas, quedarían más de 1 millón de elementos entrenantes. Teniendo en cuenta las características del ambiente en el que se realizarían los experimentos, era necesario reducir dichos datos manteniendo las características del conjunto original y la presencia de todas las relaciones.

Suponiendo que se deseara un corpus de tamaño k, se consideraron varios métodos para la obtención de un corpus adecuado para el uso de entrenamiento y prueba de nuestro modelo. La primera idea era seleccionar k elementos aleatorios del conjunto original de tripletas. Este método de selección no garantizaba la presencia de todas las relaciones en el corpus final, por lo que fue desechada. Luego se pensó en ordenar las palabras en orden descendente de acuerdo a la cantidad de tripletas en las que aparece y seleccionar las tripletas que contienen a alguna de las p primeras palabras de la lista. Como se menciona anteriormente, las relaciones son mayormente entre mismas PoS, por lo que teniendo en cuenta también la cantidad de tripletas que contienen las relaciones *causal* y *entailment*, si seguíamos dicho criterio de selección de palabras, terminaríamos con una sobrepoblación de sinónimos, y probablmente una probre presencia, quizás nula, de las demás

relaciones.

Una mejor opción sería aplicar el método anteriormente descrito, pero en vez de escoger las tripletas asociadas a las p palabras con más ocurrencias del conjunto entero, hacer dicha selección individual para cada relación. Dicho de otra manera, ver para cada palabra, en cuántas tripletas aparece de cada relación, y luego escoger las tripletas asociadas a las selccionadas. Esta opción garantiza la presencia de todas las relaciones, sin embargo la distribución de tripletas por relación no se mantendría igual que en el conjunto original. Este problema desaparece si en vez de seleccionar las p primeras palabras de cada relación se escoge un porcentaje de las tripletas asociadas a cada relación aleatoriamente, así se garantiza que la proporción de la cantidad de palabras seleccionadas concuerde con la distribución de palabras por relación en el conjunto original. Luego se genera el ruido a partir del conjunto seleccionado, y juntos formarán el corpus de elementos entrenantes.

El porciento de selección va en dependencia del tamaño que se desee en el corpus. Durante la etapa experimental se usaron conjuntos con el 10 %, 25 %, 50 %, 75 % y 100 % de los datos originales. Con ellos se entrenó el modelo propuesto siguiendo distintas configuraciones y teniendo en cuenta las características del dispositivo en que se realizaría la experimentación.

Para la fase de experimentación se usó el *training set* construído a partir del 50 % de los datos originales (Tabla 3.1). Se usó el 80 % del conjunto para el aprendizaje y el resto para la etapa de *testing*. Se entrenó con 3 *epochs* y un *batch size* de 32, mostrando un comportamiento que se puede ver en la Fig 3.2.

Relaciones	WordNet	Ruido	**Total**
Sinónimos	152297	151955	304252
Antónimos	3552	3454	7006
Hipérnimos	43389	43332	86721
Hipónimos	43389	43565	86954
Holónimos	10749	10828	21577
Merónimos	10749	10983	21732
Causal	44	43	87
Entailment	199	192	391

Tabla 3.1: Contenido del corpus (50 %)

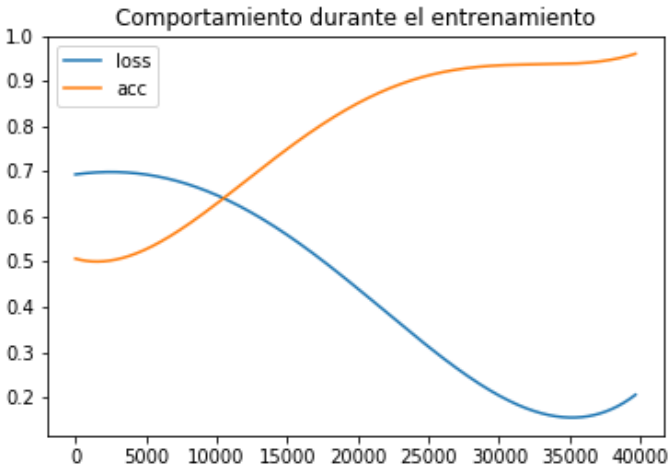


Figura 3.2: Comportamiento durante el entrenamiento

3.3. Embeddings resultantes

Como se explica anteriormente, el modelo contiene dos embeddings, uno para las palabras y otro para las relaciones de WordNet. A continuación se analizan las propiedades y características que mostraron dichos embeddings entrenados.

3.3.1. Palabras

Una de las primeras características que se encontró fue la relación entre la correctitud de una tripleta con la distancia entre sus palabras. Para esto se calculó para cada 4-tupla (w_1, r, w_2, v) del corpus la distancia entre los vectores de las palabras w_1 y w_2, las cuales se encontraban en el intervalo $(0, 4)$. Además, se separaron las tripletas originales de WordNet de las que fueron generados como parte del ruido (Fig 3.3.

De los datos extraídos se puede ver una dependencia entre la veracidad de una tripleta y la distancia entre los vectores asociados a las palabras que componen dicha relación. También que la mayor parte de las distancias en tripletas verdaderas muestran valores bajos, mientras que las falsas muestran valores más altos. Esta característica se puede notar un poco más en un modelo entrenado solo con las relaciones *synononym* y *antonym* (Fig 3.4), de lo que se puede inferir que el predominio de tripletas existentes con la

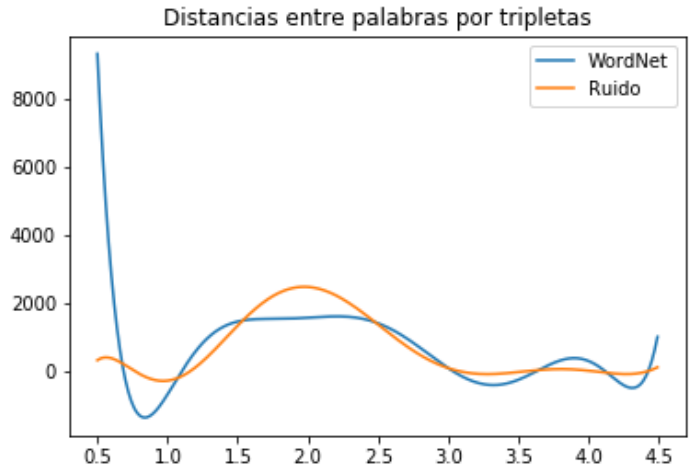


Figura 3.3: Distancias entre palabras de una tripleta

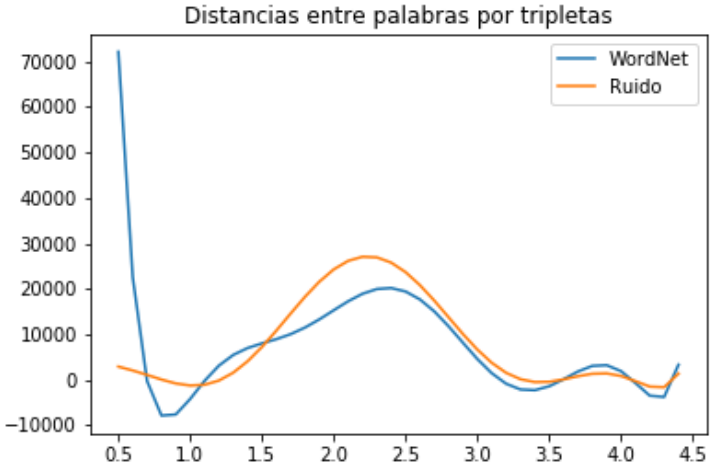


Figura 3.4: Distancias entre palabras de una tripleta (Sinónimos)

relacion de sinonimia influye en la exactitud de dicha característica.

3.3.2. Relaciones

Con el embedding de las relaciones ya entrenado, se analizaron las propiedades y características de las matrices resultantes por cada relación. Para esto se buscaron señales de las capacidades de simetría, reciprocidad y transitividad de algunas relaciones en las matrices que las representaban.

Simetría

Como se menciona en el Capítulo 2, la relación de sinonimia es simétrica, por lo que encontrar una simetría en la matriz resultante de pasar la relación de sinonimia por el embedding de relaciones daría un nivel de cuanto aprendió la propiedad de simetría de la relación. Al analizar la matriz, no se encuentra una clara simetría, por lo que se decidió buscar cuan çerca"de la simetría se encontraba la matriz. Para esto se define un çoeficiente de simetríaçalculado a partir de ella.

$$sim(M) = \frac{(M - M^T)^2}{||M|| \times ||M^T||} \tag{3.1}$$

Como se puede ver en la Tabla 3.2, los menores valores de simetría se encuentran en las dos únicas relaciones simétricas con las que fue entrenado nuestro modelo. También se puede ver que el resto de las relaciones muestran coeficientes de simetría bastante altos comparados con las dos de menor valor. Esta característica muestra una tendencia de nuestro modelo a aprender la propiedad de simetría de las dos relaciones simétricas consideradas de WordNet.

Reciprocidad

Otra característica que puede ser interesante investigar, es la relación lógica que poseen *hypernomy/hiponomy* y *holonomy/meronomy*. Se puede llamar relaciones *recíprocas* a estos pares de relaciones. Una relación r_1 es recíproca de una relación r_2, si para toda tripleta $(w_1 r_1 w_2)$ existente en el corpus, la tripleta $(w_2 r_2 w_1)$ pertenece también a este. Para analizar si el embedding aprendió la reciprocidad de las relaciones, se puede definir como similaridad de una matriz M_1 con la tranpuesta de otra matriz M_2 como:

$$tran(M_1, M_2) = \frac{(M_1 - M_2^T)^2}{||M_1|| \times ||M_2^T||} \tag{3.2}$$

Relaciones	sim(M)
Sinónimos	**0.12**
Antónimos	**0.65**
Hipérnimos	1.97
Hipónimos	2.04
Holónimos	2.00
Merónimos	2.07
Causal	1.92
Entailment	1.81

Tabla 3.2: Coeficientes de simetría por relación (menor valor = mayor simetría).

Para realizar las comparaciones, además de las parejas de relaciones recíprocas conocidas, se agruparon las cuatro relaciones restantes de dos parejas. Como se puede ver en la Tabla 3.3 las relaciones que poseen la propiedad de reciprocidad muestran valores pequeños, mientras que las otras parejas no recíprocas tienen valores mayores. Con esto se puede deducir que el modelo aprende cuales relaciones tienen dicha propiedad.

Relaciones	tran(M)
Sinónimos / Antónimos	2.31
Hipérnimos / Hipónimos	**0.22**
Holónimos / Merónimos	**0.39**
Causal / Entailment	2.02

Tabla 3.3: Coeficiente de similaridad con su tranpuesta por relación.

Transitividad

La transitivdad es una propiedad que poseen muchas de las relaciones de WordNet. A continuación se analiza la capacidad del modelo de aprender dicha característica en las relaciones con más presencia de datos en el corpus que la poseen.

Para comprobar la capacidad del modelo de aprender la transitividad de una relación r se seleccinaron 100 parejas de palabras (w_1, w_2) tal que las tripletas (w_1, r, w) y (w, r, w_2) exisitieran en el corpus. Luego se ejecutó

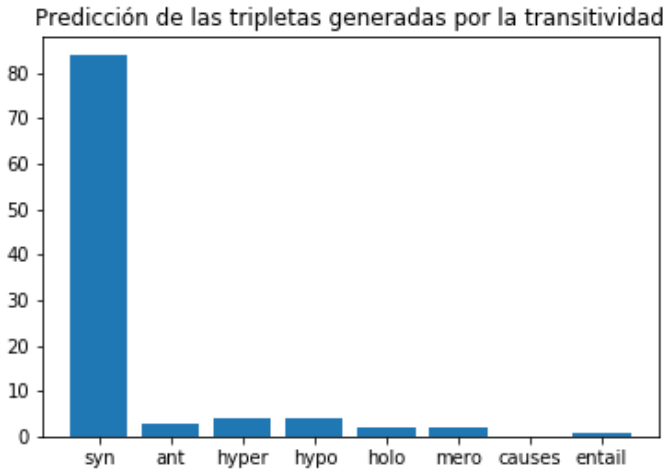


Figura 3.5: Cantidad de pares de palabras ganadoras por relación (Sinónimos)

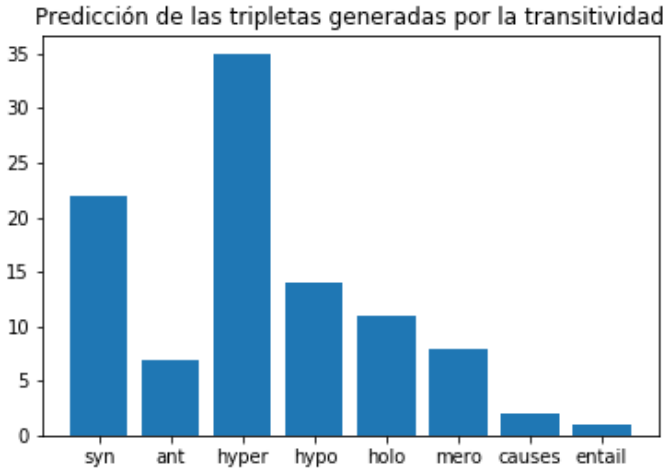


Figura 3.6: Cantidad de pares de palabras ganadoras por relación (Hipérnimos)

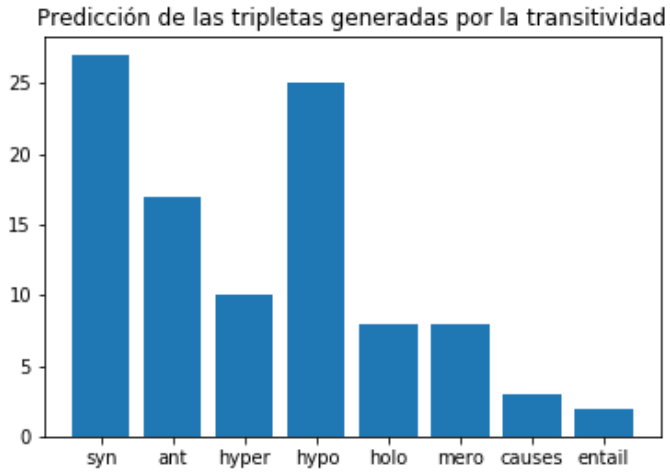


Figura 3.7: Cantidad de pares de palabras ganadoras por relación (Hipónimos)

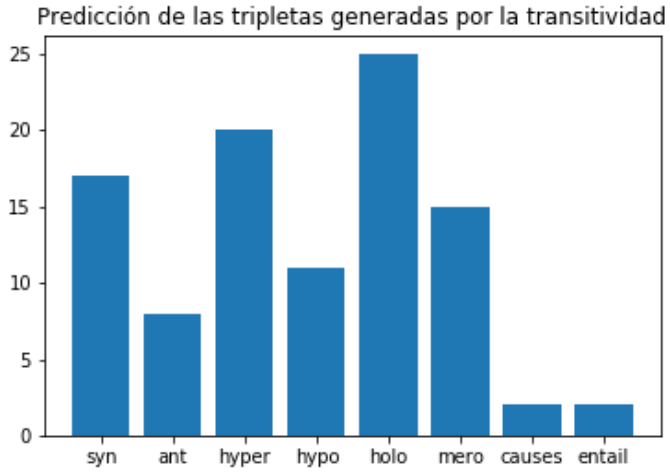


Figura 3.8: Cantidad de pares de palabras ganadoras por relación (Holónimos)

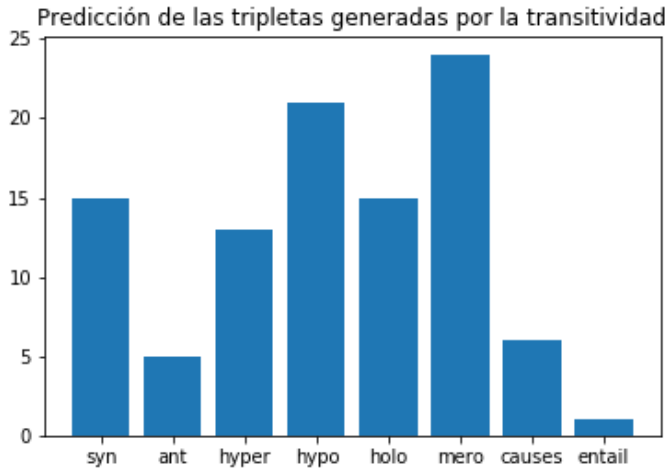


Figura 3.9: Cantidad de pares de palabras ganadoras por relación (Merónimos)

el modelo con las 8 tripletas posibles por cada pareja de palabras, una por cada relación. Se considera que el modelo aprendió la transitividad de un par de palabras si de las 8 ejecuciones realizadas con cada relación, la tripleta (w_1, r, w_2) es la de mejor resultado luego de su ejecución. Entonces, la efectividad del modelo en el aprendizaje de la transitividad se define como el porcentaje de parejas de palabras de la muestra seleccionada que en sus 8 ejecuciones, la tripleta asociada a la relación r fue la de mejores resultados.

Como se puede ver en la Tabla 3.4, el modelo aprendió con un **84%** de efectividad la transitividad de la sinonimia. Los valores reusltantes fueron muy cercanos a 1, el cual representa el mejor valor posible. Con las demás relaciones la efectividad no fue tan alta, aunque en la mayoría de los casos, la relación que se evaluaba fue la de mejores resultados (Figs 3.5, 3.6, 3.7, 3.8, 3.9).

Esta diferencia, como se ha mencionado anteriormente, ocurre debido a la gran diferencia de tripletas de sinónimos con las demás relaciones. Quizás en un corpus con cantidades de tripletas más parejas, la transitivdad de las demás relaciones se aprendería con mayor efectividad.

Relaciones	Efectividad (%)
Sinónimos	84
Hipérnimos	35
Hipónimos	25
Holónimos	25
Merónimos	24

Tabla 3.4: Efectividad en el aprendizaje de la transitividad

3.4. Discusión

Los resultados experimentales que se muestran en este capítulo permiten analizar el comportamiento y funcionamiento de la propuesta dada. Se comprueba que la información contenida en WordNet presenta una organización bien estructurada. Pero la enorme diferencia entre la cantidad de palabras relacionadas por sinonimia y las relacionadas por alguna de las restantes dificulta el aprendizaje debido a una falta de datos. Sin embargo, el modelo propuesto es independiente de WordNet y de otras ontologías. Cualquier base del conocimiento estructurada que se pueda representar en un conjunto de términos emparejados por entidades de relaciones, puede ser usado en el aprendizaje del modelo. A pesar de las difcultades con la ausencia de datos, la propuesta aprende a detectar la veracidad de las tripletas. De esta manera, obtiene la capacidad de reconocer si dos elementos presentan o no una relación dada.

Como se pudo ver en el Capítulo 1, la mayoría de los embeddings estudiados se enfocan en las entidades en sí, y no en sus relaciones, en este caso léxicas y semánticas. El modelo presentado se enfoca en dichas relaciones con el objetivo de encontrar y aprender algunas de sus propiedades. Como resultado a dicho enfoque, aprende las características simétricas de las relaciones de sinonimia y antonimia. También aprende a reconocer la reciprocidad que poseen algunas parejas de relaciones en WordNet como los hipérnimos y los hipónimos. Además, se comprobó que aprende la transitividad de la sinonimia con excelents resultados, y aunque la falta de datos afecta al efectividad en als demás relaciones, se puede ver el aprendizaje de dicha propiedad en las demás relaciones que la poseen. Todo esto muestra la capacidad del modelo de aprender otras propiedades que conozcamos de algunas relaciones. Además, permitirá encontrar propiedades de simetría o reciprocidad a aquellas a las que no se les conocía. Incluso ayudará a encon-

trar nuevas relaciones ocultas a partir de características algebraicas entre las matrices asociadas a las relaciones y los vectores de los términos.

Conclusiones

En esta investigación se propone un modelo de aprendizaje para la representación de textos basado en *Word Embeddings* y *Entity Embeddings*. Esta propuesta se aleja de las técnicas usadas en trabajos anteriores donde el entreamiento se realiza con grandes textos planos y aprovecha los beneficios de una base del conocimiento construida a partir de una ontología como WordNet. Se realizó una profunda investigación de la literatura especializada en la representación de textos para el Procesamiento del Lenguaje Natural, determinándose así los beneficios del desarrollo de un modelo basado en entidades y en las relaciones entre estas. Se analizó la arquitectura de WordNet, sus definiciones y relaciones más importantes. Se realizó la extracción de la información estructurada contenida en en dicha ontología considerando los resultados de su análisis, para luego ser usada en el entrenamiento del modelo. Se definió un modelo capaz representar como vectores los términos o palabras, y las relaciones como matrices. De esta manera se puede aprovechar mediante sencillas operaciones algebraicas entre sus representaciones, la información que en conjunto puedan brindar ciertos términos relacionados. Se implementó el modelo definido usando las herramientas brindadas por Tensorflow y Keras. Se realizaron una serie de experimentos con el objetivo de analizar las propiedades de los *embeddings* aprendidos. Los resultados de estos experimentos muestran la capacidad del modelo de predecir la correctitud de una pareja de palabras relacionadas, así como de aprender ciertas características de dichas relaciones, como la simetría, la reciprocida y la transitividad.

Recomendaciones

La investigación realizada en la tesis puede extenderse con el uso de otras ontologías con diferentes estructuras e idiomas. El uso de una base del conocimiento con una mejor proporción en sus datos puede conllevar a un mejor aprendizaje del modelo.

Una posible forma de enriquecer los resultados expuestos en la fase de experimentación puede ser mediante el ajuste de los parámetros asociados al entrenamiento del modelo. El uso del corpus entero generado, la ejecución de más *epochs* y el aumento del *batch size* puede resultar en un aumento de la efectividad del modelo. La distribución de los datos parece tener una gran influencia en la precisión de los datos obtenidos. En consecuencia se propone la construcción de un corpus con una distribución más uniforme en cuanto a las relaciones semánticas que puedan existir.

La búsqueda de otras propiedades existentes entre las relaciones e incluso de relaciones desconocidas puede ser un buen aprovechamiento de las características encontradas durante la experimentación. Ninguna de las publicaciones revisadas enfocan su investigación en las relaciones, por lo que la profundización en ese sentido puede simplificar los métodos de representación futuros y enriquecer la semántica representada.

Los resultados de este proyecto están concebidos como parte de un trabajo más general que se desarrolla en el grupo de investigación. Por tanto es necesario vincular el modelo propuesto a los diferentes problemas de Procesamiento del Lenguaje Natural que se estudian en la actualidad, con el objetivo de comprobar su efectividad.

Referencias

[1] Alfonso, Marcel: *Influencia temática en Twitter: una propuesta basada en tópicos*, 2017. (Citado en la página 1).

[2] Almeida-Cruz, Yudivián; Estévez-Velarde, Suilan y Piad-Morffis, Alejandro: *Detección de Idioma en Twitter (Language Detection on Twitter)*. 2014. (Citado en la página 1).

[3] Bengio, Y: *Neural net language models*. 3:3881, Enero 2008. (Citado en la página 6).

[4] Bengio, Yoshua; Ducharme, Réjean; Vincent, Pascal y Janvin, Christian: *A Neural Probabilistic Language Model*. J. Mach. Learn. Res., 3:1137–1155, Marzo 2003, ISSN 1532-4435. `http://dl.acm.org/citation.cfm?id=944919.944966`. (Citado en la página 6).

[5] Blei, David M.; Ng, Andrew Y. y Jordan, Michael I.: *Latent Dirichlet Allocation*. J. Mach. Learn. Res., 3:993–1022, Marzo 2003, ISSN 1532-4435. `http://dl.acm.org/citation.cfm?id=944919.944937`. (Citado en la página 5).

[6] Brown, Peter F.; deSouza, Peter V.; Mercer, Robert L.; Pietra, Vincent J. Della y Lai, Jenifer C.: *Class-based N-gram Models of Natural Language*. Comput. Linguist., 18(4):467–479, Diciembre 1992, ISSN 0891-2017. `http://dl.acm.org/citation.cfm?id=176313.176316`. (Citado en la página 6).

[7] Camacho-Collados, José; Pilehvar, Mohammad Taher y Navigli, Roberto: *Nasari: Integrating explicit knowledge and corpus statistics for a multilingual representation of concepts and entities*. Artificial Intelligence, 240:36–64, 2016. (Citado en la página 9).

[8] Cao, Yixin; Huang, Lifu; Ji, Heng; Chen, Xu y Li, Juanzi: *Bridge text and knowledge by learning multi-prototype entity mention embedding.* En *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volumen 1, páginas 1623–1633, 2017. (Citado en la página 9).

[9] Collobert, Ronan y Weston, Jason: *A Unified Architecture for Natural Language Processing: Deep Neural Networks with Multitask Learning.* En *Proceedings of the 25th International Conference on Machine Learning*, ICML '08, páginas 160–167, New York, NY, USA, 2008. ACM, ISBN 978-1-60558-205-4. `http://doi.acm.org/10.1145/1390156.1390177`. (Citado en la página 7).

[10] Dumais, S. T.; Furnas, G. W.; Landauer, T. K.; Deerwester, S. y Harshman, R.: *Using Latent Semantic Analysis to Improve Access to Textual Information.* En *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '88, páginas 281–285, New York, NY, USA, 1988. ACM, ISBN 0-201-14237-6. `http://doi.acm.org/10.1145/57167.57214`. (Citado en la página 5).

[11] Estévez, Suilan: *Minería de opinió en Twitter: propuesta de una metodología*, 2013. (Citado en la página 1).

[12] Estévez, Suilan: *Minería de opinió en Twitter: una aproximación desde el aprendizaje supervisado*, 2015. (Citado en la página 1).

[13] Firth, JR: *Papers in Linguistics.* Oxford University Press, 1957. (Citado en la página 4).

[14] González, Laila: *Metodología para el reconocimiento de entidades nombradas en mensajes cortos*, 2017. (Citado en la página 1).

[15] Honkela, Timo: *Self-Organizing Maps In Natural Language Processing.* Diciembre 1998. (Citado en la página 5).

[16] Honkela, Timo; Pulkki, Ville y Kohonen, Teuvo: *Contextual Relations of Words in Grimm Tales, Analyzed by Self-Organizing Map*, Diciembre 1998. (Citado en la página 5).

[17] Hu, Zhiting; Huang, Poyao; Deng, Yuntian; Gao, Yingkai y Xing, Eric: *Entity Hierarchy Embedding.* páginas 1292–1300, Enero 2015. (Citado en las páginas 8 y 9).

[18] Koo, Terry; Carreras, Xavier y Collins, Michael: *Simple Semi-supervised Dependency Parsing*. Agosto 2009. (Citado en la página 6).

[19] Landauer, Thomas K; Foltz, Peter W. y Laham, Darrell: *An introduction to latent semantic analysis*. Discourse Processes, 25(2-3):259–284, 1998. https://doi.org/10.1080/01638539809545028. (Citado en la página 5).

[20] Li, Yuezhang; Zheng, Ronghuo; Tian, Tian; Hu, Zhiting; Iyer, Rahul y Sycara, Katia P.: *Joint Embedding of Hierarchical Categories and Entities for Concept Categorization and Dataless Classification*. En *COLING*, 2016. (Citado en la página 8).

[21] Liang, Percy: *Semi-supervised learning for natural language*. Tesis de Doctorado, 2005. (Citado en la página 6).

[22] Lund, Kevin y Burgess, Curt: *Producing high-dimensional semantic space from lexical co-occurence*. 28:203–208, Junio 1996. (Citado en la página 5).

[23] Lund, Kevin; Burgess, Curt y Atchley, Ruth: *Semantic and associative priming in high-dimensional semantic space*, Enero 1995. (Citado en la página 5).

[24] Martin, Sven; Liermann, Jörg y Ney, Hermann: *Algorithms for Bigram and Trigram Word Clustering*. Speech Commun., 24(1):19–37, Abril 1998, ISSN 0167-6393. http://dx.doi.org/10.1016/S0167-6393(97)00062-9. (Citado en la página 6).

[25] Mikolov, Tomas; Chen, Kai; Corrado, Greg y Dean, Jeffrey: *Efficient estimation of word representations in vector space*. arXiv preprint arXiv:1301.3781, 2013. (Citado en la página 7).

[26] Mnih, Andriy y Hinton, Geoffrey: *Three New Graphical Models for Statistical Language Modelling*. En *Proceedings of the 24th International Conference on Machine Learning*, ICML '07, páginas 641–648, New York, NY, USA, 2007. ACM, ISBN 978-1-59593-793-3. http://doi.acm.org/10.1145/1273496.1273577. (Citado en la página 6).

[27] Mnih, Andriy y Hinton, Geoffrey: *A Scalable Hierarchical Distributed Language Model*. En *Proceedings of the 21st International Conference on Neural Information Processing Systems*, NIPS'08, páginas 1081–1088, USA, 2008. Curran Associates Inc., ISBN 978-1-6056-0-949-2.

http://dl.acm.org/citation.cfm?id=2981780.2981915. (Citado en la página 6).

[28] Morin, Frederic y Bengio, Yoshua: *Hierarchical Probabilistic Neural Network Language Model*. En *AISTATS*, 2005. (Citado en la página 7).

[29] Pereira, Fernando; Tishby, Naftali y Lee, Lillian: *Distributional Clustering of English Words*. En *Proceedings of the 31st Annual Meeting on Association for Computational Linguistics*, ACL '93, páginas 183–190, Stroudsburg, PA, USA, 1993. Association for Computational Linguistics. https://doi.org/10.3115/981574.981598. (Citado en la página 6).

[30] Quintana, Claudia: *Solución computacional analítica para la promoción de la salud y el bienestar humanos*, 2017. (Citado en la página 1).

[31] Ratinov, Lev y Roth, Dan: *Design Challenges and Misconceptions in Named Entity Recognition*. En *Proceedings of the Thirteenth Conference on Computational Natural Language Learning*, CoNLL '09, páginas 147–155, Stroudsburg, PA, USA, 2009. Association for Computational Linguistics, ISBN 978-1-932432-29-9. http://dl.acm.org/citation.cfm?id=1596374.1596399. (Citado en la página 6).

[32] Ristoski, Petar y Paulheim, Heiko: *Rdf2vec: Rdf graph embeddings for data mining*. En *International Semantic Web Conference*, páginas 498–514. Springer, 2016. (Citado en la página 8).

[33] Ritter, H y Kohonen, T: *Self-organizing semantic maps*. 61:241–254, Agosto 1989. (Citado en la página 5).

[34] Sahlgren, Magnus: *The Word-Space Model: Using distributional analysis to represent syntagmatic and paradigmatic relations between words in high-dimensional vector spaces*. Enero 2006. (Citado en la página 4).

[35] Shalaby, Walid y Zadrozny, Wlodek: *Learning Concept Embeddings for Efficient Bag-of-Concepts Densification*. arXiv preprint arXiv:1702.03342, 2017. (Citado en la página 9).

[36] Shalaby, Walid; Zadrozny, Wlodek y Jin, Hongxia: *Beyond Word Embeddings: Learning Entity and Concept Representations from Large Scale Knowledge Bases*. arXiv preprint arXiv:1801.00388, 2018. (Citado en la página 9).

[37] Suzuki, Jun; Isozaki, Hideki; Carreras, Xavier y Collins, Michael: *An Empirical Study of Semi-supervised Structured Conditional Models for Dependency Parsing*. En *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 2 - Volume 2*, EMNLP '09, páginas 551–560, Stroudsburg, PA, USA, 2009. Association for Computational Linguistics, ISBN 978-1-932432-62-6. http://dl.acm.org/citation.cfm?id=1699571.1699585. (Citado en la página 6).

[38] Turian, Joseph P.; Bengio, Yoshua y Roth, Dan: *A Preliminary Evaluation of Word Representations for Named-Entity Recognition*. 2009. (Citado en la página 7).

[39] Turney, Peter D. y Pantel, Patrick: *From Frequency to Meaning: Vector Space Models of Semantics*. J. Artif. Int. Res., 37(1):141–188, Enero 2010, ISSN 1076-9757. http://dl.acm.org/citation.cfm?id=1861751.1861756. (Citado en la página 4).

[40] Ushioda, Akira: *Hierarchical Clustering of Words*. En *Proceedings of the 16th Conference on Computational Linguistics - Volume 2*, COLING '96, páginas 1159–1162, Stroudsburg, PA, USA, 1996. Association for Computational Linguistics. https://doi.org/10.3115/993268.993390. (Citado en la página 6).

[41] Yamada, Ikuya; Shindo, Hiroyuki; Takeda, Hideaki y Takefuji, Yoshiyasu: *Joint learning of the embedding of words and entities for named entity disambiguation*. arXiv preprint arXiv:1601.01343, 2016. (Citado en la página 9).

[42] Zhao, Hai; Chen, Wenliang; Kit, Chunyu y Zhou, Guodong: *Multilingual Dependency Learning: A Huge Feature Engineering Method to Semantic Dependency Parsing*. En *Proceedings of the Thirteenth Conference on Computational Natural Language Learning: Shared Task*, CoNLL '09, páginas 55–60, Stroudsburg, PA, USA, 2009. Association for Computational Linguistics, ISBN 978-1-932432-29-9. http://dl.acm.org/citation.cfm?id=1596409.1596418. (Citado en la página 6).