Facultad De Ingeniería UDELAR

Proyecto de grado

Detección de Hiperonimia en Español con Representaciones Vectoriales de Palabras

Tutor:
Mathias, Etcheverry
Dina , Wonsever

Estudiante:
Gun Woo, Lee

Junio, 2020

Resumen

En el presente trabajo se aborda la tarea de detección de hiperonimia en español de forma supervisada. La misma ha sido investigada en diferentes idiomas pero según nuestro conocimiento hay escasos recursos para el español. Por este motivo, en este proyecto se construye un conjunto de datos para la relación de hiperonimia en español. Para la construcción se utilizó WordNet e información extraída de corpus. Para mejorar la calidad de los datos obtenidos se aplicaron heurísticas de filtrado.

Por otro lado se consideraron dos modelos de redes neuronales aplicados sobre representaciones vectoriales de las palabras: 1) modelo con concatenación de vectores y 2) modelo de *order embedding* de Vendrov et al. (2015). Ambos modelos son entrenados con el conjunto de hiperonimia construido.

Para evitar el fenómeno de memorización léxica (Levy et al., 2015) y estudiar el comportamiento de los modelos en palabras que no han sido vistas durante el entrenamiento, se consideran dos formas de partir el conjunto de datos (en entrenamiento, validación y evaluación) según ocurran o no palabras en común entre las particiones.

Como resultado se presenta un conjunto de datos para el entrenamiento supervisado de la relación de hiperonimia en español, con 106592 y 44960 instancias para los casos con y sin palabras en común, respectivamente. Los mejores resultados fueron obtenidos con el modelo de *order embedding*, dando un valor de 89,3 % y 81,8 % de medida F_1 sobre las dos variantes del conjunto de datos anteriormente mencionadas.

Palabras clave: hiperonimia, representaciones vectoriales de las palabras, redes neuronales artificiales.

Índice general

Capítulo 1

Introducción

Este proyecto de grado se enmarca en el área del Procesamiento del Lenguaje Natural (abreviado como PLN). El PLN es un campo de investigación dentro de las ciencias de la computación, inteligencia artificial y lingüística, que estudia cómo modelar computacionalmente el lenguaje humano. El objetivo del PLN es que las computadoras sean capaces de entender, interpretar y manipular el lenguaje humano. Los principales desafíos del PLN son la comprensión del lenguaje natural y la generación del lenguaje. Y dentro de estos grandes desafíos se encuentran tareas como la traducción automática y la extracción de información de textos.

La hiperonimia es la relación semántica que vincula a una determinada unidad léxica con otra de significado más general. La identificación exitosa de esta relación puede aportar mejoras significativas en tareas de PLN, entre las cuales es posible destacar:

- Búsqueda de respuestas a preguntas:
  Su objetivo principal es permitir que un usuario pueda plantear preguntas en lenguaje natural y extraer las informaciones correspondientes. Un posible uso de la hiperonimia es al procesar las entidades sustantivos o verbos de las preguntas del usuario. Por ejemplo, dada la pregunta "¿Qué es una pandemia?", la información de que "pandemia" es hipónimo de "enfermedad" es relevante para el sistema (McNamee et al., 2008).

- Extracción de la información:
  Su objetivo es extraer de forma automática información estructura-

da desde documentos legibles por una máquina. La información de la hiperonimia favorece la tarea de reconocimientos de nombres, donde se busca los elementos atómicos del texto como nombres de personas, lugares, organizaciones, etc. Por ejemplo, del texto "Luis disfruta paseando en bicicleta", el sistema debe extraer información que el nombre "Luis" refiere a una persona y es el sujeto en el texto.

- Reconocimiento de implicación textual:
  La idea de esta tarea es dado un par de oraciones, determinar si una se deduce de la otra, o sea, si cuando una es verdadera la otra también es verdadera. Por ejemplo, la oración "Esto fue visto como una traición por el EZLN y otros grupos políticos" implica "EZLN es un grupo político". Existe trabajo como el de Akhmatova (2009), donde a partir de ciertas relaciones de hiperonimia se aborda el problema de reconocimiento de la implicación textual.

Además, dentro del campo de la ingeniería lingüística, las relaciones de hiperonimia se han usado en áreas específicas como:

- En la creación de diccionarios y otros recursos de consulta léxica, cuya información proviene de repositorios textuales. Algunos trabajos representativos que abarcan esta área son el de Denicia-Carral et al. (2006), o el de Sierra et al. (2008).

- En el desarrollo de taxonomías y ontologías, cuya estructuración se basa en las relaciones de hiperonimia. Como trabajos representativos se encuentran el de Snow et al. (2006) y el de Paul Buitelaar and Magnini (2005).

En este proyecto se aborda la problemática de la detección de la hiperonimia para el idioma español. En primer lugar se construye un conjunto de datos de la relación de hiperonimia. Luego con el mismo conjunto de datos se entrena diferentes modelos neuronales con el uso de vectores de palabras previamente entrenados como entrada.

Si bien la tarea de detección de la relación de hiperonimia se ha abordado ampliamente en el idioma inglés, no hay tantos trabajos realizados en español. En especial según nuestro conocimiento hay escasos conjuntos de datos disponibles en español para entrenar los clasificadores que detectan la relación de hiperonimia basados en métodos supervisados, por lo que creemos

que el mismo proyecto es un aporte valioso para la disciplina de PLN en español. El aporte de este proyecto consiste fundamentalmente en los siguientes dos puntos:

1. Construir un corpus de relaciones de hiperonimia para el español con el fin de contribuir a la generación de recursos de PLN para el español. Se utilizaron diferentes fuentes y metodologías para lograr una calidad aceptable para su uso futuro.

2. Desarrollar técnicas para la detección automática de hiperonimia para el idioma español con métodos supervisados, tales como modelos de redes neuronales entrenados con vectores de palabras.

El resto del informe se estructura de la siguiente manera: El capítulo 2 está dedicado al marco teórico y antecedentes de la problemática, donde se introduce los conceptos principales de los temas que se tratan en el informe. En el capítulo 3 se presentan diferentes metodologías que se emplearon para la construcción del corpus y sus resultados correspondientes. En el capítulo 4 se presentan los modelos de redes neuronales construidos con sus detalles de implementación. Los resultados finales de los modelos se muestran en el capítulo 5. Por último en el capítulo 6 se indican el trabajo futuro y la conclusión.

Capítulo 2

Antecedentes

A continuación se introduce el marco teórico de la problemática y se describe los conceptos principales de los temas del informe.

2.1. Relación de hiponimia-hiperonimia

La hiponimia-hiperonimia es una relación semántica que se establece entre dos términos. Un hiperónimo designa aquel término que cuyo significado es más general (o amplio) que el del otro término, denominado hipónimo. Es decir que un hipónimo es un término más específico que posee todos los rasgos semánticos del hiperónimo, y además añade otras características semánticas que lo diferencian de este (Aura Josefina Rios Rios, 2009). Por ejemplo, el concepto tulipán (hipónimo) comparte todas las características del concepto flor (hiperónimo) y además, posee características adicionales como el hecho de que pertenezca a la familia de las liliáceas. Es decir, el hipónimo tulipán además de tener los rasgos semánticos del hiperónimo flor, pertenece a la familia de las liliáceas y esta última característica no se cumple en todas las flores. Ver la Figura 2.1.

A los distintos hipónimos de un mismo hiperónimo se les denomina cohipónimos. Por ejemplo, tulipán, rosa y geranio son hipónimos de flor y cohipónimos entre ellos. Los cohipónimos mantienen entre sí una relación de incompatibilidad respecto al hiperónimo. Es decir, la relación que mantienen no es una simple oposición de significado, sino que se excluyen entre sí. Si algo es una rosa no puede ser un tulipán. Cabe hacer énfasis en que la relación de hipe-

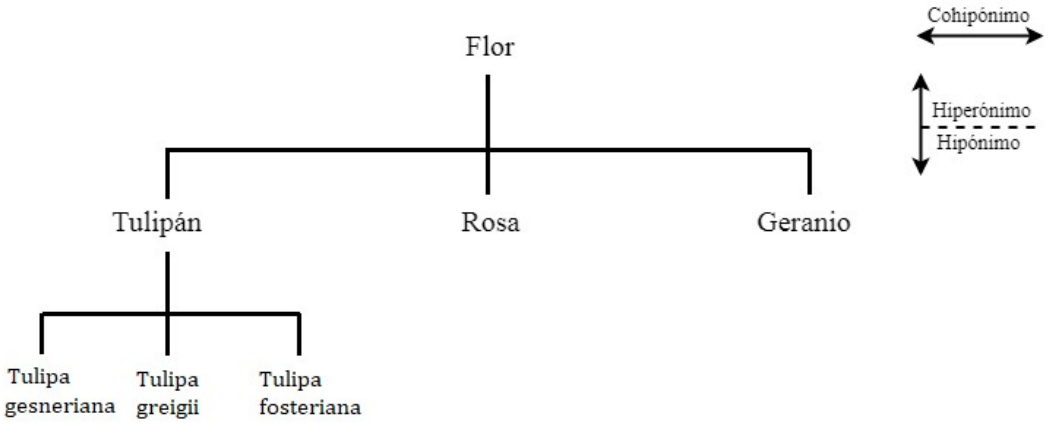


Figura 2.1: Ejemplos de la relación de hiperonimia

ronimia es una relación entre los sentidos de las palabras, de esta forma la palabra "banco" es un hipónimo de "asiento" y también de "entidad financiera", dependiendo del contexto.

Miller et al. (1991) define que existe una relación de hiperonimia entre dos conceptos X e Y si los hablantes nativos del idioma aceptan oraciones construidas a partir de frases como "Una X es un Y" o "Una X es un tipo de Y". Y explica que las relaciones de hiperonimia son básicas dentro de toda interfaz léxico-semántica de una lengua natural, debido a que una de sus funciones más importantes es estructurar sistemas de conceptos dentro de la mente de un humano, organizados conforme a las propiedades o atributos que tales conceptos prediquen de una entidad o un evento.

La relación de hiperonimia puede caracterizarse algebraicamente como una relación binaria asimétrica y transitiva. Si X es hipónimo de Y, no se cumple la inversa. Además si X es hipónimo de Y e Y es hipónimo de Z se cumple que X es hipónimo de Z. Por ejemplo, a partir de *(husky siberiano, perro)* y *(perro, animal)* es posible deducir una nueva relación *(husky siberiano, animal)*. Sin embargo, esta última propiedad no se cumple en todos los casos. Por ejemplo, dados pares de hiperonimia *(einstein, científico)* y *(científico, profesión)* es incorrecto deducir *(einstein, profesión)* ya que "einstein" es una persona y no una profesión. Existe trabajo como el de Liang et al. (2017) donde se estudia la transitividad de la relación de hiperonimia y deducir

todos los pares resultantes de la misma.

2.2. WordNet

WordNet es una gran base de datos léxica que contiene información léxica del idioma inglés. Los sustantivos, verbos, adjetivos y adverbios se agrupan en conjuntos de sinónimos llamados **synsets**, cada uno de los cuales expresa un concepto distinto. Estos synsets están enlazados por medio de relaciones léxicas y relaciones conceptuales-semánticas. La estructura general de WordNet se puede ver como una red semántica compuesta por unidades léxicas y relaciones entre ellas (Roca, 2000).

WordNet se ha venido desarrollando desde los años 80 bajo la dirección del psicolingüista George Miller en la Universidad de Princeton. La versión de WordNet en inglés es libre y se puede descargar de su página web de la Universidad de Princeton[1]. En los últimos años han sido creados WordNet individuales de diferentes idiomas los cuales enriquecieron la información léxica de otros idiomas, incrementando su uso general y aplicación en el área de estudio de lenguajes. En el día de hoy WordNet se ha convertido en un recurso estándar en todo tipo de investigaciones y aplicaciones semánticas en el área del PLN y lingüística computacional. La última versión hecha pública es WordNet 3.1 la cual consta de 117.200 entradas entre palabras y grupos lexicalizados pertenecientes a las llamadas "categorías abiertas": Nombres (69 %), Adjetivos (16 %), Verbos (12 %) y Adverbios (3 %).

2.2.1. Synset

Es la unidad básica en la que se estructura WordNet. Un synset es una agrupación de palabras que son sustituibles en al menos un contexto sin alterar el significado, como entre las palabras *caminar-andar* o *automóvil-coche*. Es decir el synset representa la acepción de las palabras. Además, un synset contiene una breve definición y, en la mayoría de los casos, una o más oraciones cortas que ilustran el uso de los miembros del synset. Si una palabra contiene varios significados entonces las distintas acepciones de la misma pertenecen a distintos synsets. Por ejemplo, la palabra "ratón" esta

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

asociada a dos synsets diferentes como se muestra en el Cuadro 2.1. Por lo tanto, el significado de cada acepción de una palabra en WordNet puede verse dado por las palabras que componen su synset y el resto de los synsets. Miller et al. (1991) destaca que el significado en WordNet es diferencial, en lugar de basado en rasgos o composicional. Es decir que el significado de un concepto viene dado por contraposición al del resto de conceptos y las relaciones entre estos.

Synsets	Semántica
Synset('mouse.n.01')	Pequeño mamífero roedor, de pelaje gris, cola fina, larga y desnuda, orejas grandes y hocico largo, que es muy prolífico y dañino.
Synset('mouse.n.04')	Aparato que poseen algunos ordenadores para controlar el cursor de la pantalla, señalar, dibujar y dar órdenes.

Cuadro 2.1: Synsets de palabra "ratón"

2.2.2. Relaciones léxicas

Así, en WordNet, las relaciones se establecen fundamentalmente entre conceptos semánticos y entre palabras léxicas. Y por medio de las mismas es posible construir una taxonomía que ordena todos los conceptos o synsets de forma jerárquica como se ilustra en la Figura 2.2. Los nodos superiores de la estructura taxonómica constituyen un conjunto de conceptos con los que cualquier entidad del modelo léxico está relacionada (entidad, abstracción, lugar, forma, estado, evento, grupo, etc). Las relaciones fundamentales que se hallan en WordNet son las que se detallan a continuación.

Sinonimia

La relación de sinonimia entre dos o más palabras se cumplen cuando existe un vínculo semántico de semejanza a partir de sus significados. En WordNet es considerado como una relación léxica básica que define la misma agrupación de los synsets. Los sinónimos manifiestan entre sí relaciones graduales de similitud, proximidad o afinidad semántica, que pueden ser absolutas, parciales o contextuales. Los sinónimos absolutos son aquellos donde

dos palabras significan exacta y rigurosamente lo mismo independiente del contexto, por ejemplo, "esposos-cónyuges". Los sinónimos parciales son aquellos que presentan una relación de proximidad relativa o imperfecta, por lo cual solo se aplica en determinado contexto, por ejemplo, "amor-cariño". El sinónimo contextual es aquel que se da en casos donde las palabras funcionan como sinónimos solo en determinados contextos de comunicación, por ejemplo, "resaca-malestar" (Coelho, 2019). Y WordNet se enfoca más en la sinonimia en contexto (parcial y contextual) como solución pragmática y realista que siempre sujeta a matices e interpretaciones, permitiendo afrontar la tarea de representar y tratar computacionalmente el conocimiento léxico-semántico de una lengua (Roca, 2000).

Hiperonimia

Es una relación semántica que se establece entre dos palabras, donde los hipónimos heredan las característica del concepto más genérico (hiperónimo) y agrega al menos una característica distintiva. (Ver la sección 2.1). En WordNet, la relación de de hiperonimia es utilizada principalmente para vincular los synsets sustantivos. La hiperonimia se caracteriza como una relación de subordinación-supraordinación y subconjunto-superconjunto, además dado que se cumple la propiedad transitiva y asimétrica, es posible construir una estructura semántica jerárquica, donde las palabras hipónimos son subordinados de las palabras hiperónimos. Ver en la Figura 2.2. Gran parte de la estructura de los sustantivos de WordNet se ha generado por medio de esta relación de hiperonimia. Esta jerarquía es limitada en cuanto a su profundidad y en la mayoría de los casos no contiene más de doce niveles de organización (Miller et al., 1991).

Meronimia

Es otra relación que vincula los synsets sustantivo. Se denomina merónimo a la palabra cuyo significado constituye una parte del significado total de otra palabra. Por ejemplo, "dedo" es un merónimo de "mano" y "mano" es un merónimo de "brazo". En WordNet esta relación es representada en tres categorías: "part-of" *(parte de)*, "member-of" *(miembro de)* y "substance-of" *(sustancia de)*. La primera de ellas relaciona a una entidad con sus componen-

tes físicos como “tronco” y “árbol”. La segunda relaciona a un conjunto con sus miembros como “árbol” y “bosque”. La última relaciona a una entidad con la sustancia de la que al menos en parte, está compuesta como “madera” y “mesa”. Como se puede ver en la Figura 2.2, si bien la relación de meronimia no forma parte de la estructura jerárquica semántica, es utilizada para referenciar y vincular diferentes synsets sustantivos (Miller et al., 1991).

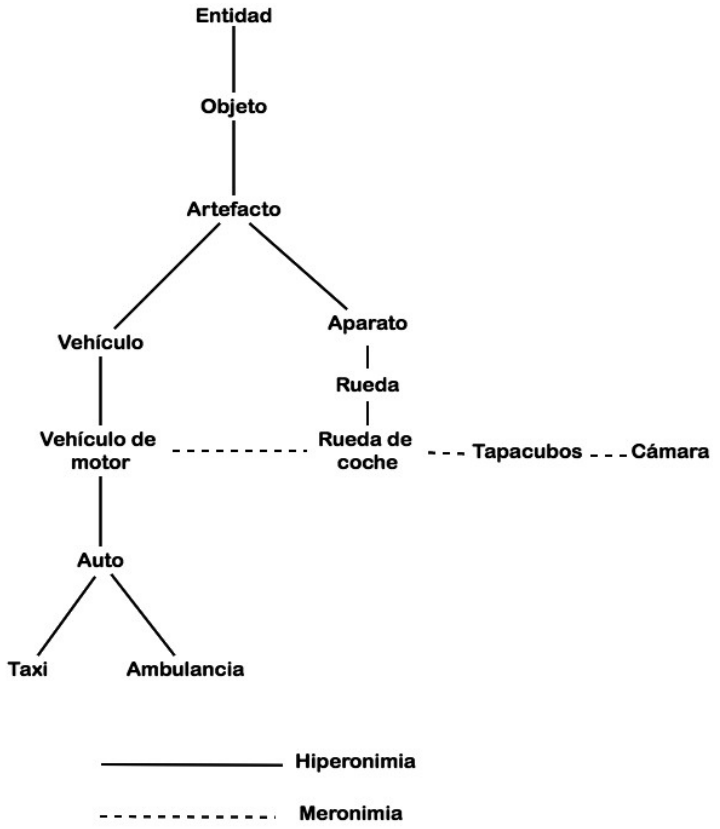


Figura 2.2: Relaciones léxicas[2]

[2]Esquema de relación extraída de Roca (2000). Cabe hacer notar que las relaciones se establecen entre los synsets y no entre las palabras. Cada palabras que aparecen en la figura representa un synset, por ejemplo, Synset(‘Entidad’), Synset(‘Auto’), etc

Antonimia y Similitud

Los antónimos son palabras que pertenecen a la misma categoría sintáctica, cuyo significado es contrario u opuesto como las palabras: grande-chico, vida-muerte. En WordNet a diferencia de las relaciones como la hiperonimia y meronimia, la antonimia se establece entre las palabras léxicas y no a nivel semántica. Existen relaciones de antonimia entre los synsets sustantivos pero no forma parte de la estructura jerárquica de los mismos. En cambio es fundamental para la organización de los adjetivos y los adverbios (Miller et al., 1991).

En cuanto a los adjetivos, WordNet clasifica en dos categorías: descriptivos y relacionales. Los adjetivos descriptivos son aquellos que adscriben a los sustantivos valores de atributos bipolares, y por tanto están organizados en base a oposiciones binarias. De esta manera los adjetivos descriptivos están organizados como pares de adjetivos que cumplen la relación de antonimia. Por ejemplo, dado "el cajón es grande". En este caso el sustantivo "cajón" tiene un atributo "tamaño" cuyo valor es "grande". Y a su vez éste atributo presenta la bipolaridad "grande - chico". Para aquellos adjetivos que no poseen antónimos directos tienen antónimos indirectos en virtud a su similitud semántica con otros adjetivos que sí poseen antónimos directos. Es decir, para aquellos adjetivos como "pequeño", "diminuto" que no cuenta con un antónimo directo se relaciona con el adjetivo "chico" según la similitud semántica, y se mantiene una relación de antonimia de forma indirecta con el adjetivo "grande". Ver la Figrura 2.3. Para esto WordNet contiene punteros entre los adjetivos que expresan el valor de un atributo y el synset de sustantivos que hace referencia al atributo en cuestión.

Por otro lado, los adjetivos relacionales son aquéllos que hacen referencia al sustantivo del cual derivan. Por ejemplo, "fraternal" se refiere a un hermano o "dental" se refiere a diente. A diferencia de los descriptivos, no están asociados a un atributo y no tienen antónimos directos. En WordNet, estos adjetivos relacionales tienen asignados punteros que hacen referencia a los sustantivos con los que están relacionados.

Además de adjetivos relacionales y descriptivos, también contiene un grupo cerrado de adjetivos llamados adjetivos modificadores de referencia (*reference-modifying adjectives*). Existe algunas situaciones donde el adjeti-

vo sólo modifica la referencia y no al referente. Por ejemplos, "El anterior presidente" o "Mi viejo amigo" (Miller et al., 1991).

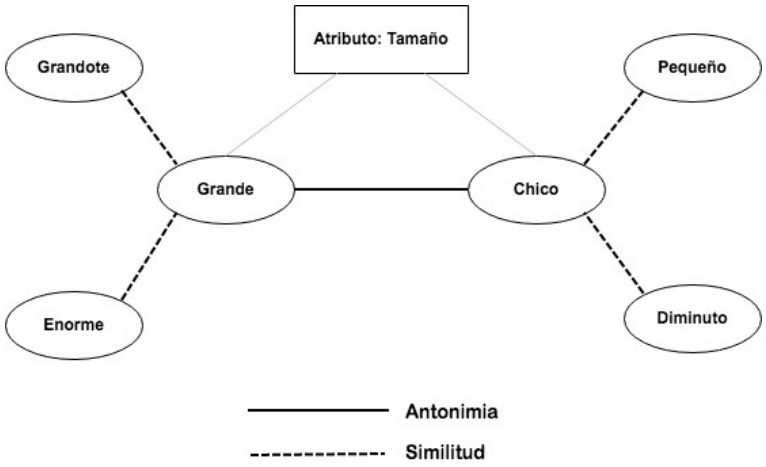


Figura 2.3: Adjetivos en Wordnet

Implicaciones

Los verbos son las categorías sintáctica más importante y compleja ya que relaciona al resto de los elementos de la frase. Los significados de los verbos son más flexibles que los de otras categorías y sus significados suelen depender de los sustantivos que lo acompañan. En WordNet no se considera como una categoría muy numerosa. Se organiza en 15 archivos diferentes, en base a criterios semánticos: cuidado corporal y fisiología, cambio, conocimiento, creación, competencia, consumo, contacto, emoción, movimiento, percepción, relación sociales, estados, comunicación, etc.

Los verbos se organizan principalmente en base la relación de consecuencia lógica, también conocido como la implicación lexical. Esta relación es una relación unilateral, es decir si el verbo V_1 implica otro verbo V_2, no puede darse el caso de que V_2 implique V_1. Por ejemplo, "roncar" implica "dormir" pero "dormir" no implica "roncar". La excepción es que cuando se puede decir que dos verbos se relacionan mutuamente, también deben ser sinónimos, es decir, deben tener el mismo sentido, como los verbos "caminar-andar". La negación invierte la dirección de implicación: "no dormir" implica "no

roncar", pero "no roncar" no implica "no dormir".

Troponimia

La **troponimia** puede verse como una hiperonimia verbal. La relación se establece entre los verbos y es considerada como un caso particular de implicación. Dado dos verbos V_1 y V_2, si V_1 es tropónimo de V_2 entonces V_1 implica V_2. Además se debe cumplir que los dos verbos ocurran simultáneamente por un tiempo. Por ejemplo, dado "cojear-caminar" o "susurrar-hablar". Los verbos en este par están relacionados por troponimia, pues "cojear" implica "caminar" y la acción de ambos verbos ocurren de forma simultanea. En contraste con pares como roncar-dormir, o comprar-pagar, están relacionados solo por implicación e inclusión temporal apropiada. Y estas actividades no ocurrir simultáneamente. Es decir uno puede dormir antes o después de los ronquidos, comprar incluye otras actividades además de pagar, por lo que ninguno de estos pares está relacionado por troponimia. Ver la Figura 2.4.

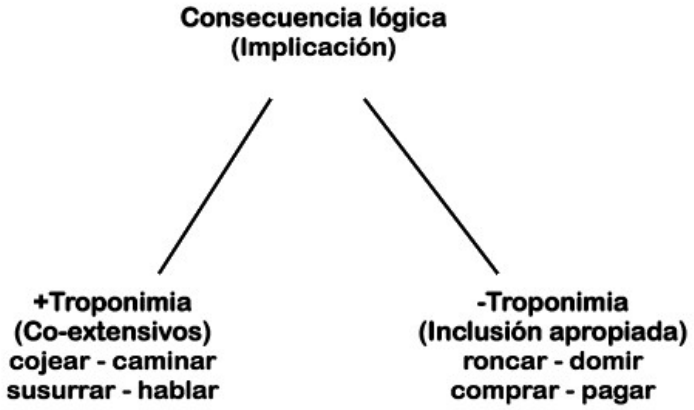


Figura 2.4: Relación de Troponimia (Miller et al., 1991)

También hay otras tipos de relaciones de implicación consideradas en WordNet como la relación de oposición y la de relación causal. Estas diferentes relaciones se ajustan mejor para organizar diferentes tipos de verbos que otros. Por ejemplo, las relaciones de troponimia, se usan en los verbos de

creación, comunicación, competición, movimiento y consumo. La relación de oposición sirve para organizar verbos de estado y verbos de cambio (Miller et al., 1991).

En cuanto a los adverbios, sólo hay pocos adverbios considerados en WordNet ya que la mayoría de los adverbios en inglés se derivan directamente de los adjetivos a través de la sufijación morfológica. Por ejemplo, "sorprendentemente", "extrañamente", etc. Y no hay estructura de árbol en la organización semántica de los adverbios como los demás. Solo se reconocen sus sinónimos y antónimos en algunos casos.

2.2.3. WordNet en otros idiomas

A pesar de que WordNet fue creado para el idioma inglés, durante los últimos años han sido creados WordNet individuales de diferentes idiomas. Diversos proyectos han desarrollado WordNet como: EuroWordNet inicialmente para holandés, italiano, español y la expansión y mejora del inglés, y en una extensión del proyecto para alemán, francés, estonio y checo; Balkanet para búlgaro, griego, rumano, serbio y turco y RusNet para el ruso, entre otros. En la página web de la Global WordNet Association[3] se puede ver una lista exhaustiva de los WordNet disponibles para diversas lenguas.

Existen dos estrategias principales para construir WordNet para lenguas distintas del inglés (Vossen, 1998).

- **Estrategia de combinación** (Merge Model): Se crea una ontología propia donde se definen las entidades, propiedades y relaciones entre ellas para la lengua objetivo. Luego con este WordNet se generan las relaciones interlingüísticas con PWN (Princeton WordNet - WordNet original). Es una técnica que requiere mayor trabajo técnico, pero permite un aprovechamiento más directo de las ontologías existentes.
- **Estrategia de expansión** (Expand Model): Se crea WordNet local de forma paralela a PWN adaptando a su red semántica. Se traduce cada synset de PWN utilizando diccionarios bilingües, traducción automática u otras estrategias. Posteriormente se revisa si las relaciones

[3] http://globalwordnet.org/

entre synsets impuestas por la estructura PWN son válidas para la lengua objetivo. Es una técnica más sencilla a nivel técnico y garantiza un grado mayor de compatibilidad entre los WordNet. La desventaja es que estos WordNet son demasiado influenciados por PWN y pueden arrastrar errores y contener algunas deficiencias estructurales.

Dentro de WordNet que fueron construidos en base a la estrategia de expansión y tienen alto nivel de cubrimiento en comparación con PWN se encuentran: MultiWordNet (Pianta et al., 2002), Multilingual Central Repository (Gonzalez-Agirre et al., 2012), WNE (Montraveta., 2010), LAS-WordNet (Jimenez and Dueñas, 2018).

Open Multilingual WordNet

A pesar de que hay numerosas WordNet individuales para diferentes idiomas, su tamaño y calidad son muy variados, tanto en su precisión como su riqueza. Además, existe escasa estandarización en términos de formato qué información contiene y su licencia. para todo usuario interesado en el tema. En la práctica, esto implica poder acceder a diferentes WordNet por medio de una interfaz común. Además OMW cuenta con una estructura de datos particular con mínimas restricciones legales que permite ampliar los WordNet ya existentes de forma gratuita o crear un nuevo WordNet la cual puede ser incorporado sobre la misma OMW.

La construcción de OMW fue basado en la estrategia de expansión y los datos de Wiktionary. Por ejemplo, dado synset $dog_{n:1}$ de PWN que representa el concepto del *perro como animal*; le asocia **lemas** (unidad que representa un sentido específico) de diferentes idiomas obtenidos de Wiktionary como *chien* en francés, *perro* en español. Específicamente en OMW el vínculo se establece entre los sentidos de los WordNet, manteniendo la jerarquía principal entre los synsets de PWN. Como los WordNet individuales se encuentran enlazados a diferentes versiones de PWN, mediante los scripts de mapeo creado por Bond and Foster (2013), fueron combinados en una única estructura multilingüe con una versión común. Debido a gran variedad de formatos en los que se distribuyen WordNet, OMW publica los scripts para cada nuevo proyecto, junto con WordNet reformateado. Bond and Foster (2013) extiende OMW combinando datos recopilados de forma automática de Wiktionary y Unicode Common Locale Data Repository (CLDR) (Daudé et al., 2003).

Como resultado, se tiene el Open WordNet Multilingüe que abarca más de 26 idiomas, que cuenta con más de 2 millones de sentidos para 117,659 conceptos, utilizando más de 1.4 millones de palabras en cientos de idiomas. Todos los datos de OMW se encuentran disponibles para descargar y también es posible acceder y manipularlos mediante la herramienta NLTK.

2.3. Redes neuronales artificiales

El cerebro es uno de las cumbres de la evolución biológica, ya que es un gran procesador de información. Entre sus características podemos destacar, que es capaz de procesar a gran velocidad numerosa información procedentes de los sentidos, combinarla o compararla con la información almacenada y dar respuestas adecuadas. Además es de destacar su capacidad para aprender a representar la información necesaria para desarrollar tales habilidades, sin instrucciones explícitas para ello.

Los científicos llevan años estudiándolo y se han desarrollado algunos modelos matemáticos que tratan de simular su comportamiento. Éstos modelos, llamados modelos de **Redes Neuronales Artificiales** (RNA) se han basado sobre los estudios de las características esenciales de las neuronas y sus conexiones.

Aunque éstos modelos son aproximaciones lejanas de las neuronas biológicas, son muy interesantes por su capacidad de aprender y asociar patrones parecidos, lo que nos permite afrontar problemas de difícil solución con la programación tradicional.

Con el paso de los años, los modelos de neuronas iniciales han ido evolucionando, introduciendo nuevos conceptos llegando a ser un paradigma de computación basado en el comportamiento de las neuronas (Ballesteros, 2006).

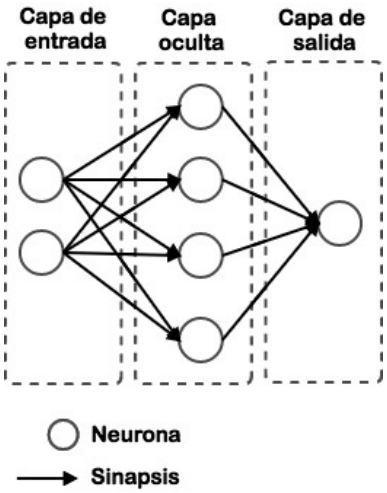


Figura 2.5: Esquema de una red neuronal simple

2.3.1. Organización y funcionamiento de las RNA

Consiste en un conjunto de unidades, llamadas neuronas artificiales, interconectadas entre sí, reciben información y la procesa y luego la envía a la siguiente neurona. Las unidades de procesamiento se organizan en capas. Hay tres partes normalmente en una red neuronal : una capa de entrada, con unidades que representan los campos de entrada; una o varias capas ocultas; y una capa de salida, con una unidad o unidades que representa el campo o los campos de destino. Los datos de entrada se presentan en la primera capa, y los valores se propagan desde cada neurona hasta las neuronas de la capa siguiente. Al final, se envía un resultado desde la capa de salida. Todas las unidades están interconectadas con otras y cada conexión, también llamado sinapsis tiene un valor de peso asignado (o ponderaciones) que modifica el valor de la salida de unidad anterior. A la salida de la neurona, puede existir una función de activación o umbral, que modifica el valor resultado o lo restringe antes de propagarse a otra neurona. Ver la Figura 2.5.

La red aprende examinando los registros individuales, generando una predicción para cada registro y realizando ajustes a las ponderaciones cuando realiza una predicción incorrecta. Este proceso se repite muchas veces durante un tiempo lo cual llamamos etapa de entrenamiento y la red sigue mejorando sus predicciones hasta haber alcanzado uno o varios criterios de parada.

La red comienza con una configuración inicial de pesos por ejemplo, peso aleatorio, y por lo tanto posiblemente su resultado contiene muchas predicciones erradas. La red aprende a través del entrenamiento. Continuamente se presentan a la red ejemplos para los que se conoce el resultado, y las respuestas que proporciona se comparan con los resultados conocidos. La información procedente de esta comparación se pasa hacia atrás a través de la red, cambiando las ponderaciones gradualmente. A medida que progresa el entrenamiento, la red se va haciendo cada vez más precisa en la replicación de resultados conocidos. Una vez entrenada, la red se puede aplicar a casos futuros en los que se desconoce el resultado (Center).

2.3.2. Neurona artificial

A continuación se detallan los elementos principales de una neuronal artificial:

- Un conjunto de entradas $x_1, ..., x_n$.
- Los pesos sinápticos $w_1, ..., w_n$ correspondientes a cada entrada.
- Una función de agregación: $\sum$
- Una función de activación: f

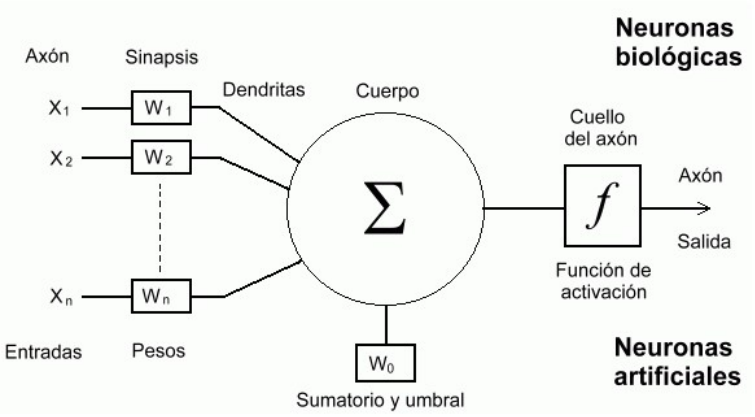
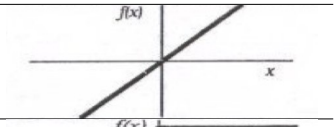
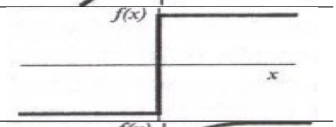
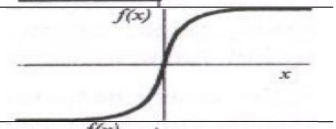
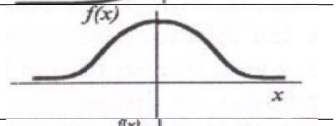
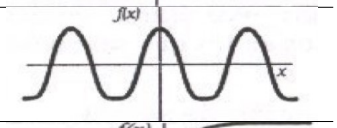
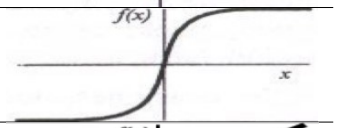
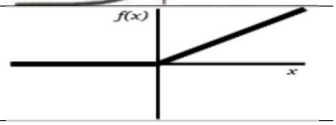


Figura 2.6: Elementos de una red neuronal

Cada neurona recibe n entradas x_i de la neurona anterior $i - esima$, ponderadas con los pesos sinápticos w_i. Con la suma de todas las entradas ponderadas genera la entrada ponderada total o "potencial postsináptico" de la neurona i. A este valor se añade un parámetro adicional θ (también representado como w_0) que resta al potencial postsináptico. Posteriormente se aplica la función de activación f. (Ver la Figura 2.6) Esta función es una simple función de matemática que dados los datos numéricos de entradas devuelve como salida un conjunto de valor en un rango determinado como $(0, 1)$ y $(-1, 1)$. En este modelo, la salida neuronal Y está dada por:

$$Y = f(\sum_{i=1}^{n} w_i x_i - \theta_i)$$

Existen diferentes funciones de activación que se eligen de acuerdo a la tarea realizada o según el diseño del modelo. Según nuestro conocimiento no existen fundamentos teóricos que den ventaja a cierta función de activación sobre las demás. En general se realizan pruebas con diferentes funciones y se elige según el desempeño del modelo y la eficiencia computacional. En el Cuadro 2.2 se muestran las funciones más conocidas. A pesar de las distintas características de las funciones como los valores de la salida, la más utilizada en el día de hoy es ReLu (Pedamonti, 2018)(Nwankpa et al., 2018).

	Función	Rango	Gráfica
Identidad	$f(x) = x$	[a,b]	
Escalón	$f(x) = sign(x)$ $f(x) = H(x)$	{-1, +1} {0, +1}	
Sigmoide	$f(x) = \frac{1}{1+e^{-x}}$ $f(x) = tgh(x)$	[0, +1] [-1, +1]	
Gaussiana	$f(x) = Ae^{-Bx^2}$	[0, +1]	
Sinusoidal	$f(x) = Asen(\omega x + \alpha)$	[-1, +1]	
Tangente hiperbólica	$f(x) = \frac{2}{1+e^{-2x}} - 1$	[-1, +1]	
ReLu	$f(x) = max(0, x)$	[a, +b]	

Cuadro 2.2: Ejemplos de funciones de activación

2.3.3. Métricas de evaluación

Las métricas más usadas para evaluar el rendimiento de modelo supervisado en tareas de clasificación son: *accuracy* - exactitud, *precision* - precisión, *recall* - exhaustividad y F_1. Dado un problema de clasificación, es posible representar el problema mediante una matriz de confusión. Por ejemplo, para el caso de clasificación binaria se representa como la figura 2.3, donde $\_N$ / $\_P$ (*Negativo* / *Positivo*) representa la predicción del modelo y $T\_$ / $F\_$ (*Verdadero* / *Falso*, representado en T y F respectivamente) representa si la misma predicción fue correcta o no respectivamente.

Tendiendo en cuenta el cuadro 2.3, las métricas se definen de siguiente

manera:

$$precision = \frac{TP}{TP + FP}$$

La *precision* mide la proporción de identificación positivas fueron correctas. Permite evaluar la calidad del modelo.

$$recall = \frac{TP}{TP + FN}$$

El *recall* mide la proporción de positivos reales que se identificaron correctamente. Permite evaluar la cantidad de datos que el modelo es capaz de identificar.

$$F_1 = 2 \cdot \frac{precision \cdot recall}{precision + recall}$$

El valor F_1 se calcula haciendo la media armónica entre la *precision* y el *recall*. Es una medida adecuada para un escenario donde se busca un equilibrio entre las dos medidas anteriores.

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

El *accuracy* mide el porcentaje de casos que el modelo ha acertado.

		Predicción	
		Negative	Positive
Realidad	Negative	TN	FP
	Positive	FN	TP

Cuadro 2.3: Matriz de confusión

2.3.4. Problemas adecuados para utilizar una RNA

Cualquier problemas que contemplan los siguientes aspectos pueden ser resueltos por una RNA(Mitchell, 1997):

- Instancias son representadas por pares de atributo-valor.
- La salida de la función puede ser un valor o un vector, discreto o real.

- Los datos de entrenamiento pueden contener errores.
- Los datos de entrenamiento pueden contener valores que no tiene atributo.
- Largos tiempos de entrenamiento son aceptables.
- Respuesta rápido de ejecución de la red para dar una salida.

Por otro lado la red neuronal cuenta con una capacidad de construir representaciones vectoriales. Dados un conjunto finito de variables como puede ser un vocabulario, utilizando la codificación *one-hot* del dominio y una capa de embeddings. La representación *one-hot* consiste en representar los valores de un conjunto de elemento finitos mediante un espacio de vectores binarios de dimensión igual a la cardinalidad del conjunto. A cada componente le corresponde un elemento del conjunto y el vector que lo representa consiste en un vector de ceros con un uno en la posición correspondiente al elemento representado. Por otro lado, la capa de embeddings es una matriz E de tamaño $\mid V \mid \times d$ con d dimensión de los embeddings y V el vocabulario. Esta matriz es ajustada durante el entrenamiento junto al resto de los parámetros de la red, y como resultado se obtienen representaciones vectoriales para las palabras.

2.4. Representaciones vectoriales de palabras

La técnica también conocida como **word embedding** mapea palabras a un vector n-dimensional de números reales. Dicha representación tiene propiedades de agrupamiento útiles, ya que agrupa palabras que están semánticamente relacionados. Por ejemplo, las representaciones vectoriales de pares de palabras que se relacionan por sinonimia, antonimia o hiperonimia, se encuentran cerca entre ellas ya que existe un vínculo semántico. Además, al ser representaciones vectoriales, resulta un escenario muy adecuado para la consideración de modelos neuronales. Las palabras representadas como vectores numéricos son utilizadas como entrada de las redes.

Los métodos de construcción de representaciones vectoriales para las palabras se basan en las hipótesis distribucional de Harris (1954), donde se plantea la idea de que existe una correlación entre la similitud de las distribuciones

de los contextos donde ocurren las palabras[4] y la similitud semántica. Considera que el significado de una palabra está determinado por los contextos en qué se usa, y por lo tanto, el significado de una palabra puede establecerse en función del conjunto de contextos en los que aparece. Las diferencias de significado están en correlación con las diferencias de distribución (Harris, 1954). Esta idea permitió desarrollar los modelos de semántica distribucional, también conocidos como modelos de "espacio de palabras". Estos modelos construyen representaciones semánticas mediante espacios vectoriales multidimensionales.

Los métodos para la construcción de vectores de palabras se pueden clasificar en dos enfoques principales: basados en conteo y basados en predicción. Los primeros, los métodos **basados en conteo**, se basan en estadísticas de las co-ocurrencias entre palabras y contextos, y en base a las mismas construye una representación vectorial de las palabras. Por otro lado, los métodos **basados en predicción**, aplican técnicas iterativas, basadas en descenso por gradiente, para optimizar una función de pérdida y como resultado se obtienen las representaciones vectoriales.

2.4.1. Métodos basados en conteo

Dado un corpus, una palabra aparece en diferentes contextos. Por ejemplo la palabra "cuchillo" aparecen en los contextos como: "cortar" y "usar". Considerando un conjunto de contextos es posible contar la cantidad de veces que aparece una palabra en el mismo. Y según la semántica de la palabra varía el número de veces que ocurre en cada contextos. Por ejemplo la palabra "cuchillo" va aparecer menos en los contextos como: "ver" y "escuchar". Por tanto es posible representar una palabra como un vector según la cantidad de veces que aparece en determinados contextos de un corpus. Si para cada palabra del corpus obtenemos un vector, podemos agruparlos en una tabla o matriz que tendrá tantas filas como palabras tenga el corpus. El número de columnas varia según el tipo de contexto que se considere. Cada fila es un vector que tiene codificada numéricamente la co-ocurrencia de la palabra con todos los contextos considerados, que son las columnas.

[4]La distribución de un elemento A se define como la suma de todos sus contextos (o "environments" como define Harris). Y cada contexto es una matriz de los elementos co-ocurrentes de A, cada uno en su posición particular (Harris, 1954).

Por ejemplo, es posible formar una matriz de co-ocurrencia M, donde cada fila representa un vector $X_{palabra}$ que describe el uso de la palabra en diferentes contextos del corpus (Ver el Cuadro 2.4).

	get	see	use	hear	eat	kill
knife	51	20	84	0	3	0
cat	52	58	4	4	6	26
dog	**115**	**83**	**10**	**42**	**33**	**17**
boat	59	39	23	4	0	0
cup	98	14	6	2	1	0
pig	12	17	3	2	9	27
banana	11	2	2	0	18	0

Cuadro 2.4: Matriz de co-ocurrencia M[5]simplificada

Dado el vector X_{dog}, la palabra "dog" ocurre [115, 83, 10, 42, 33, 17] veces en los contextos de [get, see, use, hear, eat, kill] respectivamente. Al considerar dos dimensiones de contextos "get" y "use", es posible ilustrar los vectores como la Figura 2.7.

[5]Ejemplo propuesto por Baroni en *Courses and Tutorials on DSM* `http://wordspace.collocations.de/doku.php/course:start`

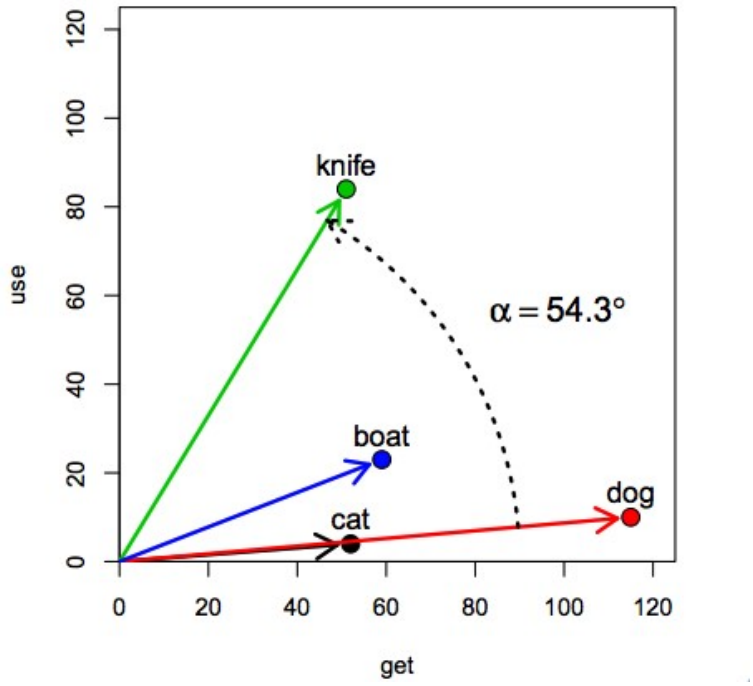


Figura 2.7: Representación gráfica de la matriz M^6

Esto permite obtener una representación de la semántica de una palabra en términos de representaciones como los vectores. Además gracias a esta representación en el espacio vectorial, es posible comparar los vectores de palabras y obtener el grado de similitud que hay entre ellos, teniendo que las palabras que aparecen en contextos similares tienen asignadas representaciones vectoriales similares. Para calcular el grado de similitud se aplican diferentes medidas, por ejemplo, la distancia coseno. Dados dos vectores de palabras en un espacio vectorial, el coseno del ángulo entre dichos vectores dará un valor entre 0 y 1. Y cuando el valor del coseno aproxima a 1, esto implica que el ángulo entre los dos vectores es pequeño es decir se asemejan las semánticas de las dos palabras. Notar que si se considera todas las palabras del corpus como contexto y una ventana de contexto simétrico (i.e. tanto el contexto izquierdo o derecho), la matriz resultante es simétrica.

La implementación de estos modelos varían según el contexto. El contexto de una palabra generalmente denota un texto en el que aparece, como un documento, una oración o un conjunto de palabras vecinas, pero también

puede contener imágenes (Feng and Lapata, 2010) o audio (Lopopolo and van Miltenburg, 2015).

Un problema de los métodos basados en conteo es que, cuando el corpus es muy grande, la construcción de la matriz de co-ocurrencia y el procesamiento de la matriz se convierten en una tarea arduo. Por lo general esto se divide en cuatro etapas (Turney and Pantel, 2010). Primero se construye la matriz de co-ocurrencia en base un determinado elemento (palabra, par de palabras, término) y la cantidad de veces que ocurrió en un determinado contexto (por ejemplo, ventana, oración o documento). Segundo, se ajusta los pesos de los elementos en la matriz, ya que las palabras comunes tendrán frecuencias altas pero son menos significativos que las palabras raras. La idea es ponderar asignando más peso a los elementos de la matriz inesperados y menos pesos a los esperados. Tercero, reducir la dimensión de la matriz buscando disminuir la redundancia manteniendo la información, por ejemplo, mediante el método de descomposición en valores singulares (Rapp, 2003). Por último calcular la similitud entre los vectores mediante algunas medidas de similitud como coseno, Jaccard, Jensen-Shannon, L1 (Weeds et al., 2004). Por ejemplo, dado x e y, la medida coseno mencionado anteriormente se calcula como:

$$x = \langle x_1, x_2, ..., x_n \rangle$$
$$y = \langle y_1, y_2, ..., y_n \rangle$$

$$\begin{aligned} cos(x, y) &= \frac{\sum_{i=1}^{n} x_i \cdot y_i}{\sqrt{\sum_{i=1}^{n} x_i^2 \cdot \sum_{i=1}^{n} y_i^2}} \\ &= \frac{x \cdot y}{\sqrt{x \cdot x} \cdot \sqrt{y \cdot y}} \\ &= \frac{x}{\|x\|} \cdot \frac{y}{\|y\|} \end{aligned}$$

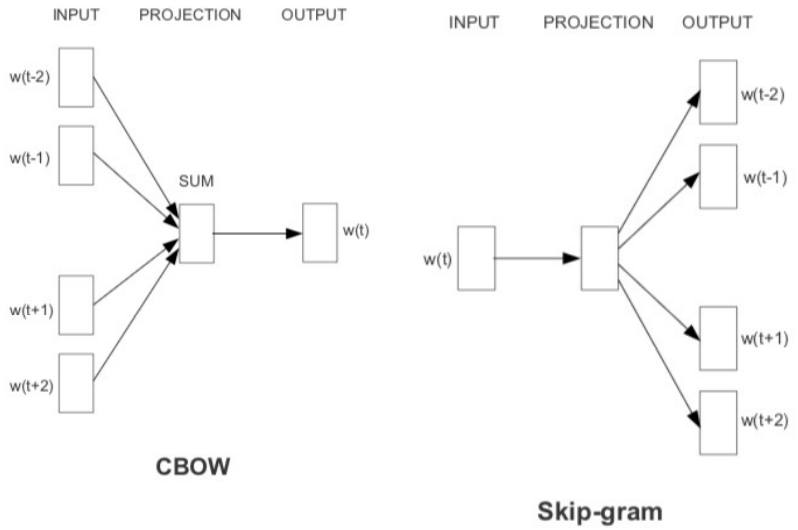
Los principales modelos en base este enfoque son el modelo LSA, también llamado como modelo de análisis latente semántico (Dumais et al., 1988), y el modelo HAL también conocido como análogo en el hiperespacio para el lenguaje (Lund et al., 1995), entre otros.

2.4.2. Métodos basados en predicción

Los métodos basados en conteo se caracterizan por que el proceso de optimización de los vectores no está supervisado y tienen consideraciones independientes basadas en la teoría de la información y el cálculo matemático (Baroni et al., 2014). Es posible enmarcar el problema de estimación de las representaciones vectoriales directamente como una tarea supervisada mediante la predicción de las palabras del contexto, donde los pesos de los vectores se configuran para predecir los contextos en los que las palabras correspondientes tienden a aparecer. A esta alternativa se le llama métodos basados en predicción. Aquí los vectores no representan directamente estadísticas sobre la distribución de una palabra y sus contextos mediante una matriz, sino indirectamente a través de la optimización de una función de pérdida, como por ejemplo, es el caso de los modelos de redes neuronales.

La idea propuesta por Hinton (1986) fue usar los pesos de las capas internas de redes neuronales para representar un concepto particular en un espacio continuo. Luego Bengio et al. (2003) plantea una arquitectura que toma vectores de palabras y una red neuronal definida sobre dicho espacio de vectores, la cual modela la probabilidad de la siguiente palabra dada una ventana de N palabras precedentes. En el trabajo posterior realizado por Tomas Mikolov et al. (2013) presenta dos nuevos modelos para aprender representaciones distribuidas de palabras y a su vez busca minimizar la complejidad computacional. Las arquitecturas de estos modelos se caracterizan por su simplicidad donde existe una única capa y no hay capa oculta como los modelos tradicionales de RNA, permitiendo que los modelos puedan ser entrenados con mayor cantidad de datos de manera eficiente.

El primer modelo se llama *Continuous Bag of Words*, más bien conocido como CBOW (ilustrado en la Figura 2.8a), define una ventana simétrica de largo fijo alrededor de una palabra central y busca optimizar la predicción de dicha palabra dado el contexto continuo. No existe la capa oculta no lineal y cuenta con una capa de proyección que se comparte todas las palabras de entrada sin importar el orden de las palabras. En el segundo modelo denominado Skipgram (ilustrado en la Figura 2.8b), busca predecir el contexto dentro de un rango definido dado una palabra de entrada.



(a) Modelo "Continuous Bag Of Words"

(b) Modelo Skipgram

Figura 2.8: Modelos de Mikolov

Como resultado lograron construir vectores de palabras de buena calidad con gran eficiencia computacional. Además, los autores muestran que los modelos vectoriales de palabras pueden responder a preguntas por relaciones de analogías por ejemplo, Francia-Paris y Alemania-Berlín, y definen un conjunto de evaluación para el inglés.

En las últimas décadas los vectores de palabras se han convertido en tipo de representación muy usada en las tareas del PLN. Actualmente existen numerosos métodos para construirlos. Dentro de los más conocidos se encuentran:

- **Word2Vec**: Los modelos de Mikolov et al. (2013) ya comentados anteriormente: CBOW y Skip-gram.
- **FastText** (Joulin et al., 2016): Es una variante del modelo Skip-gram donde cada palabra es tratada como la suma de sus subsecuencias de caracteres llamados *n-gramas*. El vector de cada palabra está compuesto por la suma de los vectores de sus n-gramas. Por ejemplo, el vector para la palabra "limón" está compuesto por la suma de los vectores para los n-gramas "<li, lim, limó, limón, imó, imón, món, ón>". Una gran

ventaja es que permite obtener mejores representaciones para palabras que tienen muy pocas o ninguna ocurrencia en el corpus.

- **GloVe**: A diferencia de los métodos anteriores, GloVe (Pennington et al., 2014) genera una matriz de conteo donde se almacena la información de la co-ocurrencia entre palabras y contextos. Cada elemento de la matriz representa la cantidad de veces que aparece una palabra en algún contexto. Luego a partir de dicha matriz se formula un problema de mínimos cuadrados de cuya solución se obtienen las representaciones vectoriales de las palabras.

Baroni et al. (2014) realiza una comparación sistemática entre modelos de predicción y conteo. Considera los problemas de similitud semántica, detección de sinonimia, categorización de conceptos y detección de analogías. Como resultado obtuvo mejor rendimiento los métodos basados en predicción en la mayoría de las pruebas realizadas.

2.5. Detección automática de hiperonimia

Dentro del área de extracción de información, se han desarrollado distintos sistemas automáticos capaces de identificar y extraer relaciones léxicas en grandes corpus e Internet. Mientras que taxonomías semánticas, como la mencionada WordNet, define relaciones de hiperonimia entre tipos de palabras, pero tienen un alcance y un dominio limitado, por lo que también resulta útil el desarrollo de métodos automáticos para identificar y extraer las relaciones de hiperonimia.

Por lo general ésta tarea se aborda utilizando dos enfoques complementarios: **método basado en patrones** y **método distribucional**. En el primero, para identificar los pares X e Y que cumplen la relación de hiperonimia, define un conjunto de patrones léxico-sintáctico que determina la relación, por ejemplo, *"X es una Y"* y busca todos aquellos pares que pertenecen a dichos patrones en un gran corpus.

En el método distribucional, la decisión de si Y es un hiperónimo de X se basa en las distribuciones independientes de los contextos de cada palabra, o en representaciones construidas a partir de ellas. Dentro del enfoque

distribucional, es posible usar métodos supervisados como no supervisados. En métodos supervisados, el par de términos X e Y está representado por un vector de características resultante de aplicar la técnica de *word embeddings* (ver la Sección 2.4.2). Estos vectores son utilizados como entrada para entrenar un clasificador binario, lo cual como salida devuelve si o no existe la relación de hiperonimia entre dos términos.

2.5.1. Métodos basados en patrones o caminos

Este método fue propuesto primera vez por Hearst (1992), quién logró identificar un conjunto de patrones léxico-sintácticos que son fáciles de reconocer, que ocurren con frecuencia en diverso género de textos los cuales indican la relación de hiperonimia. A su vez describe un procedimiento para descubrir estos tipos de patrones de forma automática en cualquier texto independiente de su idioma. En el Cuadro 2.5 se ilustran los seis patrones definidos por Hearst donde X e Y representan frase nominal.

Patrones léxico-sintácticos definidos por Hearst	
Inglés original	Traducidos en Español
"Y such as (, X)* (, and | or) X"	"Y tal(es) como (, X)* (, y | o) X"
"such Y as (X,)* (, and | or) X"	"Tal(es) Y como (X,)* (, y | o) X"
"X (, X)*, or other Y"	"X (, X)*, u otro(s) Y"
"X (, X)*, and other Y"	"X (, X)*, y otro(s) Y"
"Y including (X,)* (, y | o) X"	"Y incluyendo (X,)* (, y | o) X"
"Y especially (X,)* (, y | o) X"	"Y especialmente (X,)* (, y | o) X"

Cuadro 2.5: Patrones que indican la relación de hiperonimia

El método de Hearst se basa en patrones construidos manualmente y funciona para descubrir determinadas relaciones semánticas como la hiperonimia. Es un buen método que para su simpleza da resultados relativamente buenos. La contra que tiene es que solo detecta o encuentra pares que ocurren en el mismo contexto en el corpus.

Posteriormente Snow et al. (2005) logró construir un clasificador automático más genérico para detectar relaciones semánticas además de la hiperonimia. Snow primero colecciona los pares de sustantivos de hiperonimia

ya conocidos utilizando WordNet. Luego para cada par de sustantivos, extrae todas las frases que contienen ambos sustantivos. Cada frase extraída es compilada por un analizador sintáctico y de su árbol de derivación se obtiene su patrón léxico-sintáctico. Luego combina estos patrones utilizando un algoritmo de aprendizaje supervisado para obtener un clasificador de hiperonimia de alta precisión. A diferencia del enfoque de Hearst, Snow utiliza los caminos de dependencia para representar los patrones léxico-sintácticos. El camino de dependencia de una oración o de un patrón se genera a partir de su árbol de dependencia que representa las relaciones sintácticas entre las palabras con sus categorías gramaticales correspondientes. En otras palabras, para cada par de términos X e Y, se representan como *el conjunto múltiple de todos los caminos de dependencia que conectan a X e Y en el corpus*. Luego en base a estos caminos se entrena un clasificador para determinar si un par cumple la relación o no.

Este método identifica patrones diferentes a los de Hearst y permite obtener un mejor rendimiento. Además fue utilizado en conjunto con WordNet para la identificación de pares de hiperonimia en el corpus de *Newswire* (Snow et al., 2005).

En cuanto al español, los trabajos existentes son escasos. Ortega et al. (2011) logró identificar relación de hiperonimia a partir de una serie de patrones léxicos simplificados[6] en español. Ortega enfatiza que la construcción de patrones léxico-sintáctico no es una tarea sencilla ya que depende de herramientas lingüísticas como analizador sintáctico, etiquetador morfosintáctico, etc. Y también se debe contemplar el nivel de generalización de cada patrones ya que cuando el nivel de generalización de un patrón es más alto la confiabilidad tiende a bajar. Es decir cuando un patrón sea más genérico sintácticamente, en los pares extraídos por dicho patrón pueden aparecer más pares incorrectos, es decir, falsos positivos.

En cambio, los patrones léxicos son mas específicos y no tienen una alta capacidad de extracción. Ortega trata de descubrir gran número de patrones léxicos e iterar procesos de estimación de confianza sobre los mismos. De esta manera, un par de hiperonimia será considerada pertinente si varios patrones la extraen, y de igual manera, un patrón será adecuado mientras mayor

[6]Ejemplo de un patrón léxico simplificado: los (palabras)* y otros (palabras)*.
Ejemplo de un patrón léxico-sintáctico: NP {NP,}*{,} y otros NP

número de correctas recupere.

En primer lugar descubre un conjunto de patrones de extracción a partir de un conjunto de "semillas" lo cual representa los pares base de hipónimo-hiperónimo. Luego se recupera los documentos que contienen el conjunto de las semillas y de allí se obtiene los fragmentos de textos que figuran los pares. Dado estos fragmentos, mediante un proceso de normalización y filtrado se extraen los patrones léxicos que contiene el conjunto de semillas. Después, en la segunda fase se extrae un conjunto de tuplas mediante la aplicación de los patrones extraídos sobre la Web. Algunos patrones resultantes más relevantes se muestran en el Cuadro 2.6.

Patrón	**Confianza**
el <hipónimo>es el único <hiperónimo>	1.00
el <hipónimo>es un <hiperónimo>que	0.90
que la <hipónimo>es una <hiperónimo>	0.73
la <hipónimo>es una <hiperónimo>que	0.64
el <hipónimo>es un <hiperónimo>de	0.63
de la <hipónimo>como <hiperónimo>de	0.62
la <hipónimo>es la <hiperónimo>	0.31
de <hipónimo>y <hiperónimo>	0.06

Cuadro 2.6: Patrones de Ortega con su valor de confianza

2.5.2. Métodos distribucionales

Como se mencionó en el capítulo anterior, un problema principal de los métodos basados en patrones o caminos es que las palabras deben ocurrir en el mismo contexto, de lo contrario no se puede detectar ninguna relación. Una alternativa para detectar la relación de hiperonimia son los denominados métodos distribucionales, donde se detecta la relación en función de las representaciones distribucionales independientes de cada palabra. Habitualmente, las palabras son representadas como vectores en función de la distribución de sus contextos en corpus de gran escala (ver la Sección 2.4.2).

La propiedad de similitud entre los vectores de las palabras se conserva entre los pares de la relación de hiperonimia (Fu et al., 2014b). Es decir, da-

do los pares de hiperonimia *(carpintero, obrero)* y *(payaso, actor)* se cumple que $v(obrero) - v(carpintero) \approx v(actor) - v(payaso)$. Considerando esta propiedad de los vectores, existen trabajos que abordan la tarea de detección de hiperonimia de forma supervisada, combinando los vectores *embeddings* (los vectores resultantes de word embedding), entrena el clasificador binario. Es decir, dado un conjunto de pares de la relación de hiperonimia se obtiene sus vectores *embeddings* y para cada par se aplican operación binaria como la concatenación (Baroni et al., 2012) y la diferencia (Roller et al., 2014; Fu et al., 2014a; Weeds et al., 2014); luego con el vector resultante entrena el clasificador. Por otro lado, existe propuesta como el de Fu et al. (2014b) y de Ustalov et al. (2017) donde se aborda el problema de detección con el aprendizaje proyectado. En lugar de clasificar los pares de hiperonimia, buscan un mapeo para proyectar los embeddings de hipónimos en sus respectivos hiperónimos. En otro trabajo relacionado como el de Dash et al. (2019), se diseña una nueva arquitectura de red neuronal para asegurar la propiedad asimétrica y transitiva de hiperonimia mediante la composición de no-linealidad y conexiones residuales.

No obstante, según Levy et al. (2015) los métodos supervisados no aprenden a reconocer la relación léxica entre dos palabras X e Y. Por ejemplo, estos métodos aprenden que Y es un *hiperónimo prototípico* independiente de X, en lugar de aprender la relación explícita entre X e Y. Este fenómeno también llamado como **memorización léxica**, ocurre cuando el clasificador aprende una palabra específica como un indicador fuerte de la etiqueta. Por ejemplo, si el clasificador observa *(perro, animal), (gato, animal) y (vaca, animal)*, todos anotados como tuplas positivas, puede aprender que la palabra *"animal";* sea un hiperónimo prototípico, clasificando cualquier tupla nuevo *(X, animal)* como positivo, independientemente de la relación semántica entre ellos. Por otro lado es posible que los métodos basados en características contextuales de palabras simples no aprenden relación léxica debido a que los contextos no ofrece suficiente información para deducir la relación léxica entre dos palabras.

2.5.3. Métodos híbridos

Existen propuestas que utilizan ambos enfoques: el basado en caminos y el distribucional, para abordar la detección de hiperonimia. Shwartz et al.

(2016) considera los vectores de las palabras en conjunto con una representación de los caminos entre ambas. Esta última se establece mediante la representación de LSTM[7] (Hochreiter and Schmidhuber, 1997) de todos los caminos de dependencia que conectan a cada par de términos. Este enfoque obtuvo resultados superiores a los trabajos previos, al momento de su publicación, en comparación a los enfoques únicamente distribucionales o basados en caminos. El trabajo de Shwartz introduce además un extenso conjunto de datos para la detección supervisada de hiperonimia. Para la elaboración de este conjunto se extraen pares de hiperonimia de WordNet (Fellbaum, 1998), DBPedia (Auer et al., 2007), Wikidata (Vrandecic, 2012) y YAGO (Suchanek et al., 2007).

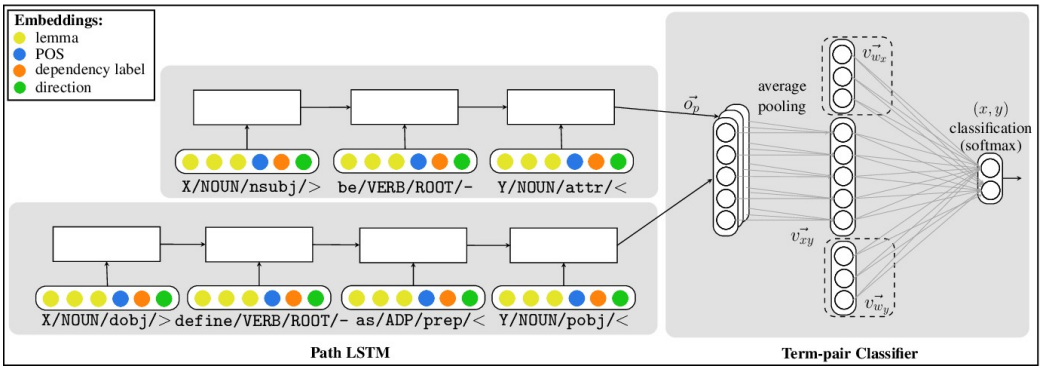


Figura 2.9: Modelo de Shwartz et al. (2016). Cada par de términos están representados por un conjunto de caminos. Cada camino es una secuencia de nodos con los siguientes elementos: lema, categoría gramatical (POS), etiqutado de dependencia, dirección. Estos caminos son tomados por LSTM y se obtienen sus vectores embeddings resultantes $\vec{o_p}$. Se utiliza el vector promediado de los embeddings para la clasificación.

[7]LSTM refiere a los modelos neuronales recurrentes *Long-short Term Memory*.

Capítulo 3

Construcción del *dataset* en español

En esta sección se describe la construcción del primer prototipo de dataset. El **dataset** consiste en un conjunto de tuplas de tres elementos donde los primeros dos son palabras del español, y el último un booleano cuyo valor es *True* o *False* si entre las dos palabras se cumple o no la relación de hiperonimia, respectivamente. Las tuplas que contienen dos palabras que cumplen la relación de hiperonimia son llamadas **instancias positivas** y se anotan *(hipónimo, hiperónimo, True)*. Por ejemplo, *(perro, animal, True)* y *(martillo, herramienta, True)*. Y a la agrupación de las instancias positivas la llamamos **dataset positivo**. Las demás tuplas que no cumplen la relación de hiperonimia son llamadas **instancias negativas** y su agrupación **dataset negativo**. En el informe para simplificar se abrevia la anotación de las tuplas con los dos primeros elementos: *(hipónimo, hiperónimo)*.

Para la construcción se utilizó Python y el *toolkit* NLTK (*Natural Language Toolkit*) ya que consideramos que es la herramienta más práctica y completa, al momento de realizar este trabajo, para el manejo de OMW y el procesamiento de datos léxicos.

3.1. Extracción de pares de hiperonimia

Un problema fundamental para la construcción de instancias positivas es que hay escasos recursos para obtener pares de hiperonimia en español. La

extracción se basó en diferentes fuentes y metodologías. En primer lugar se buscó extraer la mayor cantidad de instancias positivas desde OMW, a partir de las relaciones de hiperonimia que se establecen entre los synsets de WordNet. En segundo lugar se extrajo utilizando los patrones léxicos-sintácticos. Para esto se utilizaron los patrones sintácticos en español definidos por Ortega et al. (2011). En tercer lugar, se obtuvo una versión traducida de un sub-conjunto de las instancias de Shwartz et al. (2016). Además, se consideraron los pares obtenidos a partir de los arcos transitivos. Finalmente, se evalúa cada conjunto obtenido de instancias positivas y se emplean diferentes medidas y heurísticas para refinarlos.

3.1.1. Extracción a partir de WordNet

Como se mencionó anteriormente WordNet es una base de datos léxica construida inicialmente para el idioma inglés. El synset es la unidad básica de WordNet que representa un concepto a través de un conjunto de lemas. Por medio del OMW es posible obtener una traducción de estos lemas al español. Por tanto en primer lugar a partir de WordNet se obtiene todos los pares de synsets que cumplen la relación de hiperonimia. Luego por medio de OMW se obtiene los lemas correspondientes al español para cada uno de los synsets. Suponiendo que hay N y M lemas para un par de synset que están vinculados por la relación de hiperonimia, se generan entonces N x M pares relacionando todos los lemas entre ambos synsets (Ver la Figura 3.1). Si bien este método de extracción y su implementación fue sencilla, su resultado no fue favorable ya que las primeras instancias resultantes contenían varios errores. Posteriormente se realizó un proceso de refinamiento para mejorar la calidad de las mismas aplicando ciertas heurísticas, las cuales se detallan en la sección 3.3.

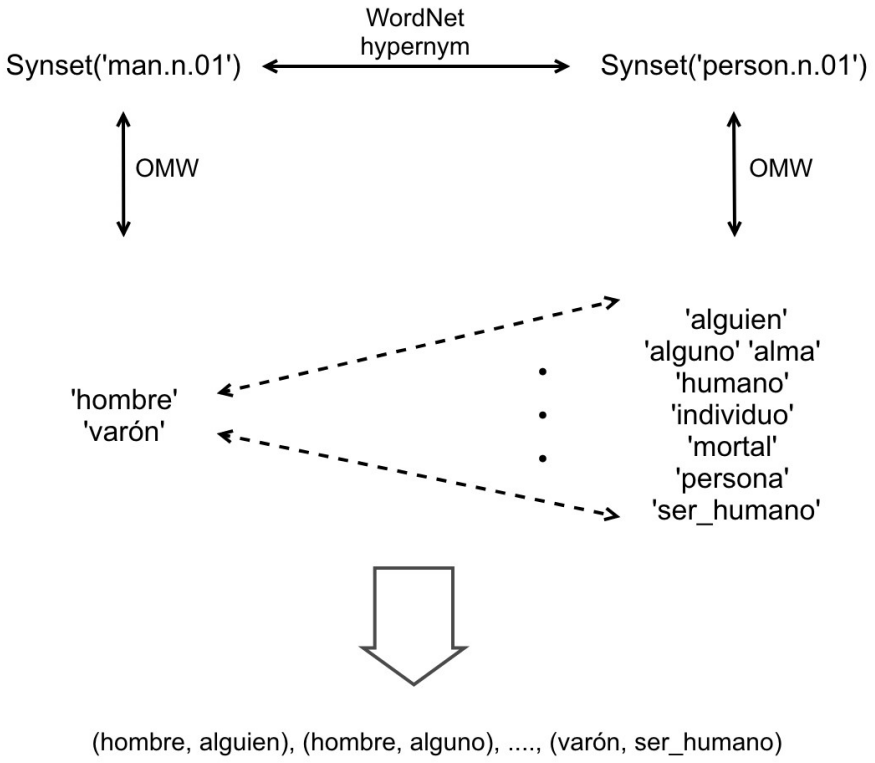


Figura 3.1: Esquema de extracción de WordNet: Dado un par de synset, Synset('man.n.01') y Synset('person.n.01'), que cumple la relación de hiperonimia, se obtiene sus lemas correspondientes en Español por medio de OMW. Luego se forman instancias positivas relacionando los dos conjuntos de lemmas.

3.1.2. Extracción basada en patrones

Considerando un conjunto de patrones léxicos-sintácticos en español es posible extraer las tuplas positivas a partir de un corpus. Basándose en el trabajo de Ortega et al. (2011), se consideraron los siguientes patrones léxicos-sintácticos en español con sus valores de confianza que se aproximan a 1:

- "el <hipónimo> es el único <hiperónimo>"

- "de <hipónimo> y otras <hiperónimo>"

Dado que estos patrones dieron buenos resultados en un dominio restringido, estudiamos el comportamiento de los mismos sobre un corpus de mayor tamaño, para analizar la calidad de las instancias resultantes. Para esto, mediante el uso de las expresiones regulares, se extrajeron pares de hiperonimia del corpus de Cardellino (2019). Posteriormente las tuplas extraídas se filtran considerando únicamente los sustantivos para mejorar su calidad.

3.1.3. Traducción del dataset de Shwartz

Como se mencionó en la sección 2.5.3, en el trabajo de Shwartz et al. (2016), los pares de hiperonimia se extraen a partir de múltiples fuentes. Su dataset contiene una cantidad considerable de instancias de ciudades, pueblos, compañías, entre otros; por ejemplo: (sydney, city), (microsoft, company). Considerando solamente aquellos pares cuyos hipónimos son nombres propios que no tienen su traducción correspondiente en español, es posible generar tuplas traduciendo únicamente la palabra correspondiente al hiperónimo. Por ejemplo, dado la tupla *(Shakespeare, writer)*, traducir solo el hiperónimo y formar la tupla *(Shakespeare, escritor)*. Limitamos nuestra selección a las instancias de: "pueblo", "ciudad", "compañía", "lugar", "río" y "persona". Para la traducción se utilizó la librería googletrans de Google[1].

3.1.4. Extracción a partir de arcos transitivos

La relación de hiperonimia cumple la propiedad transitiva, por lo tanto, es posible considerar los enlaces transitivos como tuplas positivas. Por ejemplo, dadas las tuplas *(perro, animal)* y *(animal, ser_vivo)* es posible extraer el par transitivo *(perro, ser_vivo)*. Pero a la hora de incluir sobre el dataset positivo se tomó en cuenta el problema de ambigüedad de la propiedad transitiva, mencionado en la Sección 2.1. Puesto que los arcos transitivos pueden introducir errores como ocurre en el siguiente caso: dados los pares de hiperonimia *(einstein, científico)* y *(científico, profesión)* el par resultante de la transitividad, *(einstein, profesión)*, no satisface la relación de hiperonimia. Notar que la palabra "científico" esta siendo utilizado en dos sentidos diferentes: en el primer caso se refiere a una persona que es científico y en el segundo a

[1](unofficial) Googletrans: Free and Unlimited Google translate API for Python: `https://github.com/ssut/py-googletrans`

la profesión de ser científico. A pesar de este tipo de problemas introducido por los arcos transitivos, fueron considerados para estudiar la calidad de las tuplas transitivas

Para obtener las tuplas transitivas, en primer lugar se genera un grafo dirigido acíclico con el dataset positivos ya extraído anteriormente. Luego se obtiene la clausura transitiva del grafo, sobre la cual se restan los arcos resultantes de la reducción transitiva del grafo original. Como resultado se obtienen los arcos transitivos que no estaban incluidos en el grafo original.

3.2. Construcción de instancias negativas

Los pares que no cumplen la relación de hiperonimia fueron construidos mediante la unión de cuatro subconjuntos de tuplas diferentes, los cuales se detallan a continuación.

3.2.1. Tuplas de dataset positivas invertidas

La relación de hiperonimia es asimétrica. Por tanto si una palabra X es hiperónimo de palabra Y, Y no es hiperónimo de X. Entonces teniendo ya el dataset positivo, es posible construir parte del dataset negativo intercambiando el orden de los pares positivos.

3.2.2. Cohipónimos

Como ya se mencionó en la sección 2.1, la relación de cohiponimia se cumple entre dos palabras que son hipónimos a un hiperónimo común, y además no presentan una relación de hiperonimia entre ellas. Por ejemplo, *perro*, *gato*, *ratón* son cohipónimos entre ellos con respecto al hiperónimo *animal*. Y entre ellos no se cumple la relación de hiperonimia. Por lo tanto, dado el conjunto de hiperonimia es posible obtener los pares negativos formados por los cohipónimos.

Para obtener las tuplas cohipónimos, en primer lugar se genera un grafo dirigido acíclico con el dataset positivos. Al grafo resultante se aplica reducción transitiva. Luego para cada nodo se obtiene una lista de nodos de hijos

directos. Combinando los elementos de esta última lista se genera los pares de cohipónimos.

s

3.2.3. Antónimos

La relación de antonimia se cumple entre dos palabras cuando pertenecen a la misma categoría sintáctica, cuyo significado es opuesto. Por tanto, si existe una relación de antonimia entre dos palabras no se cumple la relación de hiperonimia. Si bien en WordNet la relación de antonimia entre los sustantivos no es considerada como una relación principal, es posible extraer un conjunto reducido de tuplas de antonimia en español mediante OMW e incluirlo en el dataset negativo.

3.2.4. Tuplas aleatorias

Una forma simple de obtener instancias negativas es seleccionar dos sustantivos de forma aleatoria a partir de un vocabulario en español. En primera instancia se construyó un conjunto de sustantivos con los mismos vocabularios del dataset positivos obtenidos de OMW. Posteriormente para aumentar la cantidad de los vocabularios, se expandió el conjunto agregando los sustantivos obtenidos del corpus de Cardellino. Para esto se utilizó la librería Spacy de Python para seleccionar solo las palabras cuyo etiquetado gramatical es sustantivo. Además se consideró solo aquellas palabras que tienen más de 4 caracteres con una frecuencia mayor o igual a 200. En total se obtuvo 71,308 sustantivos en español.

3.3. Refinamiento del corpus

Luego de construir la primera versión del dataset, se evaluó su calidad. Para esto se extrajo una muestra de cada conjunto del dataset positivos y negativos, y se verificó cada tupla de forma manual si cumplía o no la relación de hiperonimia entre los pares. Sobre aquel dataset extraídos que presentaban una calidad muy inferior, se aplicó alguna medida heurística y filtrado para mejorar su calidad. A continuación se detallan los problemas

que deterioraban la calidad del dataset y las soluciones implementadas para resolver.

3.3.1. Para los pares extraídos de WordNet

La precisión de la primera versión del dataset extraído de WordNet no llegó a superar el 50 %. De los pares extraídos menos de la mitad cumplían la relación de hiperonimia. El problema mayor estaba en los errores que se introducían al traducir los pares de origen inglés a español mediante OMW. Seguidamente se detallan los problemas del dataset y las soluciones implementadas.

Orden de la lista de lemas en español

La precisión se deterioraba debido a la ambigüedad de los lemas. Pues, dado que cada synset tiene un conjunto de lemas que cada uno representa una palabra con un sentido específico, la misma palabra denota el significado del synset relacionado sobre cierto contexto determinado. Es decir dentro del conjunto de lemas de un synset pueden aparecer aquellas palabras que si bien concuerda la semántica, su uso es menos frecuente en los textos. Por ejemplo, el synset "person.n.01" y sus lemas correspondientes en inglés y en español se muestra en el Cuadro 3.1. Aparecen lemas como "alguien" y "alma" que son palabras menos frecuentes para referir el significado de "persona" y son usados sobre cierto contexto determinado.

Lemmas en inglés	Lemmas en español
person	alguien
individual	alguno
someone	alma
somebody	humano
mortal	indiviuo
soul	mortal
	persona
	ser_humano

Cuadro 3.1: Lemmas de "person.n.01"

Por otro lado, los lemas correspondientes en inglés están ordenadas según sus frecuencias, es decir, según la cantidad de aparición sobre los corpus.

Pero los lemas en español se encuentra ordenado alfabéticamente. Entonces la primera solución fue ordenar éstos lemas según su frecuencia en un determinado corpus. En el caso de las frecuencias de las expresiones de multipalabras se calcula como el promedio entre la frecuencia y la cantidad de palabras de expresión. Para esto, en primera instancia se extrajo la frecuencia desde el corpus español *cess_esp*[2] de 188,650 palabras. Luego para abarcar mayor variedad de vocabularios se extrajo la frecuencia desde el corpus de Cardellino que cuenta con alrededor de 1,5 billones de palabras en español. En el Cuadro 3.2 se ilustra como quedaron ordenados los lemas del synset "person.n.01" luego de realizada la ordenación mencionada.

	Lemmas en Español	Frecuencia
1	ser_humano	791022
2	persona	268487
3	humano	98949
4	alguien	61639
5	alguno	59196
6	alma	41629
7	individuo	27837
8	mortal	12789

Cuadro 3.2: Lemmas de"person.n.01" en español ordenadas

Una vez obtenido la lista de lemas ordenada según su frecuencia, se seleccionó los primeros elementos de la lista considerando solamente aquellas palabras mas frecuentes. Es decir tomar aquellos lemas que son más utilizados en diferentes contextos. Para verificar la mejora de la calidad se seleccionó solamente el primer elemento de la lista ordenada y como resultado efectivamente aumentó la precisión del dataset. Pero como la cantidad de los elementos a considerar de la lista es directamente proporcional al tamaño total del dataset, se evaluó su calidad y precisión variando la cantidad de los primeros elementos a considerar de la lista con fin de encontrar un dataset de calidad aceptable con mayor número de pares posibles.

[2]CESS-ESP Spanish Corpus: `http://universal.elra.info/product_info.php?cPath=42_43&products_id=1509`

Esta heurística permite reducir la ambigüedad en los pares positivos extraídos de WordNet, tomando solo las palabras que tienen la semántica más genérica independiente del contexto.

Filtrado según la frecuencia de hiperónimos

Otra heurística fue en base la hipótesis planteada en el trabajo de Santus et al. (2014), donde se plantea que los hiperónimos son más generales semánticamente que los hipónimos, y por lo tanto los hipónimos tienden a ocurrir en contextos menos informativos que los hiperónimos. Aplicando esta hipótesis sobre el escenario de las frecuencias de los términos, se planteó que los hiperónimos son más frecuentes que sus hipónimos. Entonces la heurística considerada fue dadas las frecuencias de las palabras obtenidos del corpus de Cardellino, filtrar los pares positivos tomando solo aquellos que cumplen que la frecuencia del hiperónimo sea mayor que la frecuencia del hipónimo. El resultado de aplicar estas heurísticas se detallan en la Sección 3.4.

3.3.2. Para la totalidad de los pares positivos

Dado que se cumple la propiedad transitiva en la relación de hiperonimia y hay ambigüedad en las palabras, es posible que se presenten ciclos en el dataset positivo. Los ciclos dan lugar a casos donde la relación es simétrica, introduciendo errores. Para evitar esto, se procedió detectar los ciclos sobre todo el dataset extraído y eliminarlos.

Por otro lado, detectamos la existencia de ciertas palabras frecuentes como "género_de_", "familia_", etc, las cuales observamos que deterioran el resultado del entrenamiento de los modelos. Por lo cual, se procedió a eliminarlas de todos los pares del dataset.

Eliminación de ciclos

Para tratar estos casos se utilizó Networkx[3], una librería de Python, obteniéndose las aristas involucradas en ciclos, se quita una aleatoria y se procede a verificar si todavía existen ciclos en el dataset, continuando dicho proceso

[3]Networkx es una librería de Python para la creación, manipulación y estudio de la estructura, dinámica y funciones de redes o grafos complejos. `https://networkx.github.io/`

hasta que no queden ciclos.

Cabe observar los casos de ciclos con el largo igual a dos donde si bien cumple la relación de hiperonimia en ambas direcciones dependiendo del contexto, se aproxima más a una relación sinonimia. Ver el Cuadro 3.3. Estos casos se eliminaron para reducir la ambigüedad en el dataset. Otra alternativa para tratar estos casos fue incluir ambos casos como instancias positivas y considerar la relación de hiperonimia antisimétrica, donde en el caso de igualdad se asigna como los sinónimos. Pero se descartó ya que el número de casos era bajo.

Hipónimo	Hiperónimo
golpe	toque
suspensión	pausa
retraso	pausa
tranquilidad	alivio
cumbre	cima
segundo	instante
huida	fuga
esclavo	siervo

Cuadro 3.3: Casos de Loops de largo igual a dos

Normalización del dataset

Las palabras frecuentes que detectamos que deterioran el resultado de los modelos fueron: "subgénero_de_", "género_de_", "subgénero_", "género_", "subfamilia_", "familia_", "orden_", "suborden_" y "ser_". Estas palabras fueron eliminadas en todo el dataset. Por último, todas las palabras fueron llevadas a minúsculas.

3.4. Evaluación

Para verificar la calidad del dataset, se realizó un muestreo extrayendo de forma aleatoria un conjunto de tuplas del dataset positivos y negativos. Luego comprobamos cada una de ellas si cumple la relación de hiperonimia

o no. La evaluación manual se realizó en base las definiciones previstas por *The Free Dictionary - Farlex*[4] y *Wiktionary*[5]

3.4.1. Muestreo de dataset

A continuación se muestra algunas tuplas extraídas del dataset positivos y negativo, y también se detallan los problemas que causan la ambigüedad en el mismo.

Hipónimo	Hiperónimo	Positivo
demandante	persona	correcto
síntoma	índice	correcto
tabulador	tecla	correcto
género leontocebus	mamíferos	correcto
disco	salón de baile	correcto
berkelio	elemento metálico	correcto
reglamento	norma	correcto
carnicero	mercader	correcto
átomo	fragmento	incorrecto
servidor	ordenador	correcto
paseo	viaje	correcto
dispositivo	utilaje	correcto
consanguineidad	relación	correcto
...	...	...

Cuadro 3.4: Ejemplos de tuplas extraídas del dataset positivos

[4] `https://es.thefreedictionary.com/`

[5] Diccionario libre con más de 900,000 entradas en español `https://www.wiktionary.org/`

Hipónimo	Hiperónimo	Negativo
redondez	caproidae	correcto
subdiácono	superlativo	correcto
hippoglossus	sistema operativo	correcto
regalo de casamiento	barba	correcto
álcali	ratonera	correcto
editorialista	sardina	correcto
vibración	Manidae	correcto
hallazgo fortuito	salsa marrón	correcto
dominatrix	complemento directo	correcto
doble	cactaceae	correcto
diplopía	sacrilegio	correcto
maltratador	gemfibrozil	correcto
milicia	denisonia	correcto
...	...	...

Cuadro 3.5: Ejemplos de tuplas extraídas del dataset negativo

En el caso del dataset positivo, como ya se mencionó en la Sección 3.4, la aparición de los errores en su mayoría es debido a la ambigüedad introducida en los lemas de español de OMW (por ejemplo, átomo-fragmento). Pero cabe observar que son ambiguos donde si bien no se cumple la relación de hiperonimia de forma estricta, sí se presentan cierto vínculo semántico entre ellos.

También el error ocurre a causa de la traducción misma entre WordNet y OMW. Por ejemplo, el synset *'edification.n.01'*, se refiere al concepto de la mejora de la mente y la comprensión, especialmente mediante el aprendizaje[6]; más la palabra traducida *'edificación'* no contiene el mismo concepto en español.

3.4.2. Resultado de evaluación

Los pares extraídos de WordNet se anotan con WN-NaN, donde se considera N primeros elementos de la lista de lemas luego de aplicar la primera heurística como ya se explicó en la Sección 3.3. Como se puede observar en el Cuadro 3.6, al aplicar la segunda heurística se disminuye la cantidad de

[6]Traducido de la definición de Cambridge Dictionary `https://dictionary.cambridge.org/es/diccionario/ingles/edification`

los pares totales pero aumenta su calidad considerablemente como el caso de $N = 3$.

N	Tamaño (# pares)	% Precisión
1	15695 / 10103	83.9 / 84.3
2	29180 / 19258	82.2 / 83.3
3	35103 / 22851	77.6 / 83.5

Cuadro 3.6: Se muestra el resultado al aplicar la segunda heurística (derecha) y el resultado sin aplicarla (izquierda).

La calidad de los muestreos de los demás dataset se muestra en el Cuadro 3.7. Para el muestreo de esta sección se extrajo $0,5\,\%$. El dataset extraído de los arcos transitivos dio un resultado muy inferior y consideramos que es debido al problema de ambigüedad como se mencionó anteriormente. En cuanto a las tuplas negativas formadas con los positivos invertidos, decidimos excluirlas del dataset, debido a que consideramos que podían afectar negativamente en la calidad del dataset en casos donde los pares de hipónimo e hiperónimo tienen un significado muy similar, por lo que podría aumentar la ambigüedad.

Luego de evaluar los resultados del dataset, se definió como el dataset base conformados en parte del positivo WN-3a3 aplicado ambas heurísticas y Shwartz y en parte del negativo, las tuplas obtenidos de forma aleatoria y los antónimos. Los demás dataset serán considerados como combinación con el dataset base para evaluar el aprendizaje del modelo más adelante.

Positivos	Shwartz	Pattern-based	Transitivos
Precisión	95 %	60 %	20 %
Cantidad	3798	2731	1604334

Negativos	Invertidos	Aleatoria	Cohipónimos	Antónimos
Precisión	95 %	100 %	100 %	90 %
Cantidad	$\sim$ # WN-NaN	300051	150000	1111

Cuadro 3.7: Resultado del muestreo

3.5. Partición del corpus

En esta sección se detalla como se dividió el conjunto de datos (instancias positivas y negativas) para el entrenamiento del modelo. Como es usual, la partición se realiza en tres conjuntos: entrenamiento, validación y test. Además, siguiendo el trabajo de Shwartz et al. (2016) se consideraron dos formas de partición diferentes: aleatoria y lexical. En la siguientes secciones se detallan cada una.

3.5.1. Partición aleatoria

El método también conocido como *random split* consiste en dividir el dataset de forma aleatoria con una distribución uniforme entre el dataset positivo y negativo. Es un método eficiente y simple de implementar. La proporción de los conjuntos de entrenamiento, validación y test fueron: 70 %, 5 %, 25 % aproximadamente. La proporción de random split se muestra en el Cuadro 3.8.

3.5.2. Partición sin intersección léxica

El problema que puede presentar en el proceso de entrenamiento del modelo es memorización léxica (Levy et al., 2015), ya mencionado en la sección 2.5.2. Para evitar este fenómeno, se realiza partición léxica donde se divide los conjuntos con intersección de términos nula. En otras palabras, cada conjunto contiene un vocabulario distinto. Además, el entrenamiento de un modelo en un dataset dividido sin intersección léxica, puede dar como resultado un modelo más general, que presente un mejor desempeño sobre las tuplas con términos que no aparecen durante el entrenamiento. Para esto se utilizó la implementación provista por Shwartz et al. (2016), donde se enfoca dividir los tres conjuntos, entrenamiento, validación y test con una proporción aproximada de 70/5/25 %. Las proporciones obtenidas en la partición léxica se muestra en el Cuadro 3.8.

		Train	Val	Test	Total
Random Split	Positivo	18654	1332	6662	106592
	Negativo	55962	3996	19986	
Lexical Split	Positivo	8221	513	2506	44960
	Negativo	24663	1539	7518	

Cuadro 3.8: Tamaño de las particiones

Capítulo 4

Modelos de detección de hiperonimia

El objetivo de esta sección es construir modelos de redes neuronales para predecir la relación de hiperonimia entre dos palabras en español. En primer lugar utilizando la técnica word embeddings se transforma los pares de palabras a su representación vectorial. Luego los vectores resultantes de la transformación son tomados como entrada para los modelos. En este trabajo se consideraron los siguientes modelos neuronales:

1. El modelo de concatenación de vectores *(x ⊕ y)* basado en el trabajo de Baroni et al. (2012). Este enfoque consiste en realizar una clasificación binaria sobre la concatenación de los embeddings del par de palabras del cual se quiere detectar la relación. Este enfoque se describe en la Sección 4.1.

2. El modelo basado en *order embedding* de Vendrov et al. (2015), donde se transforma el embedding de cada palabra mediante una red neuronal entrenada de forma supervisada, a una nueva representación vectorial con una relación de orden parcial establecida entre los vectores. Este enfoque se describe en la Sección 4.2.

Utilizando las particiones del dataset comentados en la sección 3.5 y sus vectores correspondientes, se entrenan los modelos anteriores sobre el conjunto de entrenamiento y se ajustan los hiperparámetros buscando una buena configuración en función de los resultados obtenidos sobre el conjunto de validación y se selecciona el mejor de los modelos entrenados. Luego se evalúa

el mejor de los modelos obtenidos según el conjunto de validación sobre el conjunto de prueba. En la figura 4.1 se muestra un esquema de los modelos.

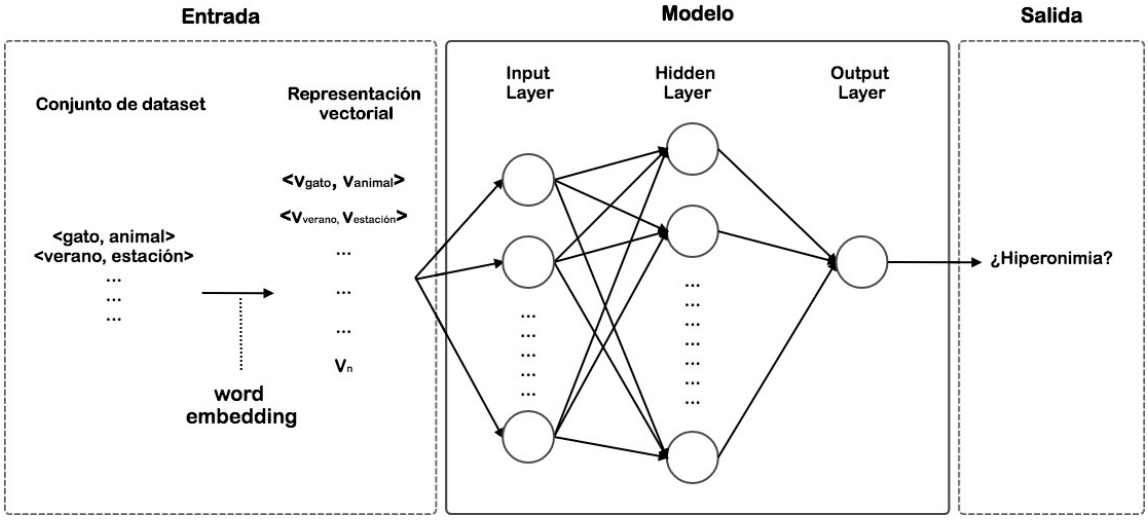


Figura 4.1: Esquema del modelo distribucional considerado para la detección de hiperonimia. A los pares de hiperonimia del dataset se aplica word embedding y se obtiene la representación vectorial de cada palabra. Estos embeddings son utilizados como entrada para entrenar los modelos.

A continuación se presentan los modelos propuestos, los detalles de implementación y los procedimientos de entrenamiento y evaluación.

4.1. Modelo con la concatenación de vectores

En esta línea se siguió la propuesta de Baroni et al. (2012), donde se concatenan los dos embeddings de cada par de palabras correspondiente al dataset, para luego suministrárselo como entrada a una red neuronal. Cada vector de palabra obtenido por fasttext tiene dimensión 300, por lo tanto, como el resultado de la concatenación de par de palabras $x \oplus y$ se tiene un vector de dimensión 600. Este vector resultante es la entrada al modelo RNA de concatenación. Y como la salida del modelo se tendrá 1 ó 0 que indica Verdadero ó Falso si se cumple la relación de hiperonimia respectivamente. La arquitectura del modelo es en capas. Para la salida se utilizó la función de activación sigmoide y como función de pérdida se utilizó *binary cross entropy*.

La configuración de los demás hiperparámetros como la cantidad capas, la cantidad de neuronas por capa, función de activación y estrategia de optimización, entre otros, se detallan en la sección 4.3.2.

4.2. Modelo *Order Embedding*

Es el modelo propuesto por Vendrov et al. (2015), donde a diferencia del modelo de concatenación, en lugar de combinar los dos vectores de palabras y entrenar un clasificador sobre el resultado, se aprende un *order embedding* en $\Re^m_{\geq 0}$ considerando el *reversed product order* que se define como:

$$x \preceq y \iff \bigwedge_{i=1}^{m} x_i \geq y_i, \tag{4.1}$$

Donde $x, y \in \Re^m_{\geq 0}$, y x_i e y_i corresponden al componente i-ésimo de x e y, respectivamente. Por definición, esta relación es antisimétrica, transitiva y $\vec{0}$ es el elemento superior de la jerarquía. Esta relación de orden parcial $(\preceq, \Re^m_{\geq 0})$ permite definir la siguiente medida para cuantificar el grado en que un par de dos elementos no satisface la relación. Consideremos

$$E_p(\vec{x}, \vec{y}) = ||max(\vec{0}, \vec{y} - \vec{x})||^2, \tag{4.2}$$

donde $\vec{x}, \vec{y} \in \Re^m_+$ y max es el máximo componente a componente. Notar que E_p indica el grado de no satisfacción de la relación y que $E_p(x, y) = 0$ si $\vec{x} \preceq \vec{y}$. Además, E_p es ajustada para ser mayor que un umbral α para términos no relacionados a través de la función *max-margin loss*:

$$E_n(\vec{x}, \vec{y}) = max\{0, \alpha - E_p(\vec{x}, \vec{y})\}. \tag{4.3}$$

Notar que $E_n(\vec{x'}, \vec{y'})$ es *0* cuando $E_p(\vec{x'}, \vec{y'}) \geq \alpha$ garantizando que no se cumpla $\vec{x'} \preceq \vec{y'}$. Luego, considerando (4.2) y (4.3), la función de pérdida resultante es la siguiente:

$$L = \sum_{(x,y)\in P} E_p(\vec{x}, \vec{y}) + \sum_{(x',y')\in N} E_n(\vec{x'}, \vec{y'}), \tag{4.4}$$

donde P y N son conjuntos de ejemplos positivos y negativos, respectivamente. Notar que L es diferenciable, lo cual permite entrenar order embeddings a través de descenso por gradiente.

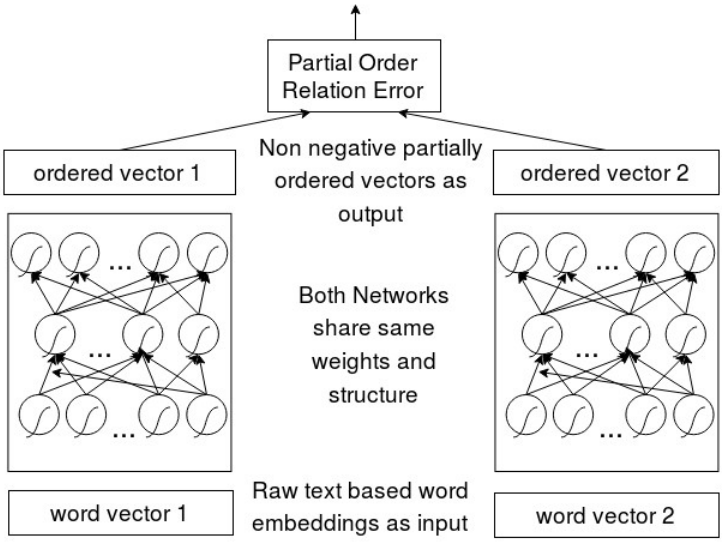


Figura 4.2: Modelo de order embedding aplicado a la detección de hiperonimia. La imagen fue extraída de Lee et al. (2020)

Esta función de pérdida es utilizada con una red neuronal para el mapeo de vectores de palabras a vectores que están ordenados mediante la relación de orden parcial que se establece en función de ejemplos durante el entrenamiento (ver la figura 4.2).

4.3. Implementación y entrenamiento

Para la implementación de RNAs se utilizó la librería Keras (Chollet), la cual proporciona un API de redes neuronales de alto nivel, escrita en Python y capaz de ejecutarse sobre TensorFlow (Google), CNTK (Microsoft) o Theano (MILA). Keras permite realizar prototipos de forma fácil y rápida, a través de la facilidad de uso, modularidad y extensibilidad. Utilizamos TensorFlow como *backend* de Keras.

Los modelos de redes neuronales involucran una cantidad considerable de hiperparámetros en su definición; como cantidad y tamaño de capas, funciones de activación de las neuronas, tamaño de *batch*, estrategia de optimización y cantidad de épocas, entre otros; con lo cual fue necesario recurrir a técnicas de búsquedas de hiperparámetros para lograr una buena combinación. Mas

adelante abordaremos este punto.

Para evaluar el desempeño de los modelos entrenados se utilizaron las métricas usuales de clasificación binaria: *precision*, *recall* y F_1 (ver la Sección 2.3.3). Los resultados obtenidos se detallan en el capítulo 5.

4.3.1. Word embeddings

Se consideraron dos recursos de word embeddings en español. Por un lado, Cardellino (2019) donde además de ofrecer un enorme tamaño de corpus cuenta con un conjunto de vectores de palabras ya entrenados con la técnica de *word2vec* (Sección 2.5.2). Por otro lado, *fastText* de Common Crawl[1] ofrece vectores de palabras pre-entrenados para distintos lenguajes, donde los vectores son entrenados sobre los corpus de Wikipedia (Bojanowski et al., 2017); (Grave et al., 2018). Nosotros hemos optado por utilizar *fastText* ya que la misma se caracteriza por su cobertura amplia sobre cualquier vocabulario de idioma.

4.3.2. Búsqueda de hiperparámetros

Los hiperparámetros definen características del modelo y el entrenamiento; y se pretende encontrar una buena configuración en términos del desempeño del modelo. Los hiperparámetros influyen en la capacidad del modelo y características del aprendizaje. Para encontrar una configuración adecuada de hiperparámetros se consideraron los métodos **manual search**, **grid search** y **random search** (Bergstra and Bengio, 2012).

Manual search refiere a la búsqueda manual de hiperparámetros, donde se establece una configuración inicial y se procede a realizar variaciones y a manetenerlas o descartarlas dependiendo de los resultados obtenidos. *Grid search* consiste en buscar exhaustivamente a través de un conjunto predefinido de valores para cada hiperparámetro. En otras palabras, se evalúa el modelo para todas las combinaciones posibles sobre el conjunto definido. Si bien es una técnica que permite encontrar la configuración óptima dentro de las evaluadas, el tiempo de procesamiento es exponencial en función de

[1]Common Crawl es una organización sin fines de lucro que rastrea la web y pone a disposición del público los datos resultantes. `https://fasttext.cc/`

la cantidad de hiperparámetros. *Random search* busca en un subconjunto especificado de hiperparámetros aleatoriamente en lugar de exhaustivamente como es el caso de grid search. Esta variante reduce considerablemente el tiempo de procesamiento. Un parámetro adicional importante para especificar en este método es *n_iter*. Esto especifica el número de combinaciones para intentar aleatoriamente. La selección de un número demasiado bajo de *n_iter* disminuye las posibilidades de encontrar la mejor combinación. Seleccionar un número demasiado grande aumenta el tiempo de procesamiento.

Los hiperparámetros que se consideraron para ambos modelos fueron:

- Cantidad de capas ocultas
- Cantidad de neuronas de las capas ocultas
- Función de activación de cada capa
- Estrategia de optimización
- Cantidad de épocas (número de iteraciones del entrenamiento)
- Batch size (número de muestras para la actualización de gradiente)

Modelo con concatenación de vectores

Inicialmente para entrenar el modelo se probaron varias configuraciones de forma manual variando algunos de los parámetros especificados anteriormente. Luego utilizando la función GridSearchCV de la librería sklearn[2], se intentó encontrar los hiperparámetros para el modelo. Sin embargo, nos encontramos con problemas debido a grandes demoras de tiempos de ejecución (más de 5 días sin lograr finalizar). Por este motivo se optó utilizar random search.

Respecto a random search, se consideraron 150 combinaciones. A continuación se detalla la configuración de los hiperparámetros encontrados según Random Search:

[2]Scikit-learn es una biblioteca de aprendizaje automático de código abierto que admite el aprendizaje supervisado y no supervisado. También proporciona varias herramientas para el ajuste de modelos, el preprocesamiento de datos, la selección y evaluación de modelos, y muchas otras utilidades. `https://scikit-learn.org/stable/`

- Cantidad de capas ocultas = 2
- Primera capa oculta con 300 neuronas y función de activación *relu*
- Segunda capa oculta son 100 neuronas y función de activación *relu*
- Capa de salida de 1 neurona con función de activación *sigmoid*
- Función de pérdida: *binary_crossentropy*
- Optimizador: *adam*
- En el entrenamiento se realizaron *20 épocas* con *batch_size de 32*

Adicionalmente, se utilizó **EarlyStopping** (Prechelt, 2000), que consiste en detener el entrenamiento cuando una medida monitoreada deja de mejorar por ciertó número consecutivo de épocas. La medida monitoreada fue la ya mencionada F_1, y se pretende una mejora mínima de un 1 % cada 5 épocas (*min_delta = 0.01 y patience = 5*), de lo contrario el entrenamiento se detiene.

Modelo de *Order Embedding*

En el modelo de *order embedding* se aplicó directamente la técnica de random search sobre algunos de los parámetros que se consideraron más relevantes. La configuración resultante fue la siguiente:

- Cantidad de capas ocultas = 3
- Primera capa oculta con 150 neuronas y función de activación *selu*
- Segunda capa oculta son 150 neuronas y función de activación *selu*
- Tercera capa oculta son 100 neuronas y función de activación *relu*
- Optimizador: *adam* con *learning rate = 0,005*
- EarlyStopping con *patience = 3*

Capítulo 5

Resultados

En esta sección se muestran los resultados obtenidos de entrenar los modelos comentados anteriormente con el dataset en español, y se comparan con otros modelos entrenados para el inglés y se analizan los resultados.

5.1. Resultados de los modelos

Para evaluar los modelos se consideraron las métricas ya mencionadas: *precision*, *recall* y F_1. El entrenamiento del modelo se realizó con diferentes combinaciones del dataset con el fin de estudiar como varía el resultado del aprendizaje del modelo en diferentes escenarios. A continuación se muestran las combinaciones consideradas:

- Dataset base definido en la Sección 3.4.2: esta conformado por tuplas WN-3a3 aplicando las heurísticas y las tuplas traducidas de Shwartz; y como instancias negativas, las tuplas extraídas de forma aleatoria a partir de los vocabularios de WordNet y Cardellino y antónimos extraídos de WordNet. (Base)
- Al dataset base se le agrega el conjunto de cohipónimos como instancias negativas para el entrenamiento. (Base +cohyp)
- Al dataset base se le agrega las instancias positivas extraídas por patrones léxicos del corpus de Cardellino. (Base +pattern)
- Al dataset base + pattern, se le agrega el conjunto de cohipónimos como instancias negativas para el entrenamiento. (Base +pattern +cohyp)

Además se evaluaron los resultados sobre la partición de test agregando las instancias de los cohipónimos. Es decir, se evaluaron los modelos entrenados según los subconjuntos definidos anteriormente sobre un conjunto de test que por un lado contempla el dataset negativo base y por otro lado el dataset negativo agregado las instancias negativas de cohipónimos.

Modelo con concatenación de vectores

Los resultados obtenidos del modelo de concatenación se muestra en el Cuadro 5.1. La tabla superior (a) corresponde al resultado de considerar las instancias de los cohipónimos solamente en la partición de entrenamiento. La tabla inferior (b) muestra el resultado de considerar los cohipónimos en las particiones de entrenamiento y prueba.

(a)

	P_{rand}	R_{rand}	F_{rand}	P_{lex}	R_{lex}	F_{lex}
Base	0.881	0.820	0.839	0.819	0.696	0.736
Base +cohyp	**0.899**	0.797	0.835	0.804	0.692	0.728
Base +pattern	0.854	0.836	0.835	**0.857**	0.646	0.721
Base +pattern +cohyp	0.845	**0.865**	**0.846**	0.758	**0.789**	**0.761**

(b)

	P_{rand}	R_{rand}	F_{rand}	P_{lex}	R_{lex}	F_{lex}
Base	0.806	0.850	0.818	0.789	**0.818**	**0.791**
Base +cohyp	**0.874**	0.767	0.806	**0.817**	0.672	0.723
Base +pattern	0.708	**0.863**	0.766	0.790	0.786	0.776
Base +pattern +cohyp	0.827	0.841	**0.825**	0.769	0.777	0.761

Cuadro 5.1: Resultados del modelo de concatenación en el dataset español.

Modelo de Order Embedding

Los resultados obtenidos del modelo de Order Embedding se muestran en el Cuadro 5.2. Igual que el cuadro anterior la tabla superior (c) considera las instancias de los cohipónimos solamente en la partición de entrenamiento y la tabla inferior (d) considera los cohipónimos en las particiones de entrenamiento y prueba.

(c)

	P_{rand}	R_{rand}	F_{rand}	P_{lex}	R_{lex}	F_{lex}
Base	0.855	0.904	0.879	**0.823**	0.674	0.741
Base +cohyp	0.857	**0.932**	**0.893**	0.809	**0.827**	**0.818**
Base +pattern	**0.860**	0.885	0.872	0.798	0.766	0.782
Base +pattern +cohyp	0.859	0.930	0.893	0.802	0.821	0.811

(d)

	P_{rand}	R_{rand}	F_{rand}	P_{lex}	R_{lex}	F_{lex}
Base	0.719	**0.946**	0.817	0.744	0.841	**0.789**
Base +cohyp	0.847	0.869	0.858	**0.781**	0.716	0.747
Base +pattern	0.742	0.931	0.826	0.666	**0.857**	0.749
Base +pattern +cohyp	**0.848**	0.870	**0.859**	0.759	0.678	0.716

Cuadro 5.2: Resultados del modelo Order Embedding en el dataset español.

Resultados para el inglés

A modo de comparación, se adjunta los resultados de otros modelos entrenados sobre el corpus del idioma inglés. Dentro de ellos se encuentran:

- El modelo HypeNET (Shwartz et al., 2016) que fue construido de forma híbrido integrando ambos enfoques de detección de hiperonimia.
- El modelo de concatenación que dio buen resultado, reportado en el trabajo de Shwartz et al. (2016).
- El modelo S-TEAL (Wang et al., 2019) basado en *Taxonomy Enhanced Adversarial Learning* - (TEAL). Un modelo supervisado que utiliza los vectores embeddings entrenados del web y conjuntos de hiperónimos anotados manualmente.
- El modelo order embedding con la función de activación variada (Lee et al., 2020). El modelo fue probado sobre el dataset inglés de Shwartz.

5.2. Análisis de los resultados

En primer lugar, dentro de los modelos construidos obtuvo mejor desempeño el modelo de order embedding. Respecto a la partición aleatoria, tanto

(e)

	P_{rand}	R_{rand}	F_{rand}	P_{lex}	R_{lex}	F_{lex}
Best Distributional (Shwartz et al., 2016)	0.901	0.637	0.746	0.754	0.551	0.637
HypeNET Integrated (Shwartz et al., 2016)	0.913	**0.890**	**0.901**	**0.809**	0.617	0.700
S-TEAL (Wang et al., 2019)	0.780	0.860	0.830	-	-	-
OrdEmb SELU-ReLU (Lee et al., 2020)	**0.932**	0.845	0.887	0.740	**0.872**	**0.801**

Cuadro 5.3: Resultados de otros modelos sobre el dataset en inglés.

en el modelo de concatenación como en el modelo order embedding no se observa una mejora considerable al añadir las instancias positivas extraídas de patrones sobre la partición de entrenamiento. Y en algunos casos como de los cuadros (a) y (c) al agregar las instancias negativas cohipónimos sobre el conjunto de entrenamiento aumenta los *recall* y F_1. Por otro lado, en general al agregar las instancias de los cohipónimos sobre el conjunto de test no hay mejora considerable.

Respecto a la partición léxica, en especial el cuadro (c) del modelo de order embedding, al añadir los cohipónimos sobre el conjunto de entrenamiento mejora los resultados en la evaluación base junto con su cobertura. Y en el modelo de order embedding (c) y (d), al añadir las instancias de patterns solos, sin cohipónimos mejoran el *recall* a costo de bajar la precisión, lo cual es esperable, dado que se introducen nuevos casos de tuplas positivas y algunas erróneas. Como se puede ver en el cuadro (b) y (d), al incluir los cohipónimos en la evaluación puede verse que los entrenamientos con cohipónimos lleva a resultados más precisos, lo cual es esperable por añadir instancias negativas que podríamos considerar "finas". Por último cabe mencionar que hay desigualdad del tamaño entre la partición aleatoria y la partición lexical (Ver el Cuadro 3.8), que pudo haber alterado los resultados entre la partición aleatoria y léxica.

En comparación a otros modelos supervisados en inglés, el modelo de order embedding tuvieron resultados competitivos. El modelo de order embedding logró un buen resultado en comparación al modelo *Best Distributional* reportado por Shwartz y también al modelo HypeNET, lo cual combina el enfoque distribucional y basado en caminos.

5.2.1. Muestreo de las instancias clasificadas

Para observar los errores de los modelos se analizaron una parte de las instancias de falsos positivos y falsos negativos respecto al conjunto de validación. Además para corroborar el aprendizaje de los modelos se analizaron una parte de las instancias de los verdaderos positivos y verdaderos negativos. La extracción se realizó sobre el caso base respecto las particiones aleatoria y léxica.

Modelo con concatenación de vectores

Sobre la partición aleatoria se obtuvo 200 (15,0 %) instancias de falsos negativos y 194 (4,8 %) instancias de falsos positivos. Y sobre la partición léxica se obtuvo 121 (23,6 %) instancias de falsos negativos y 52 (3,4 %) de falsos positivos. A continuación se muestran algunos errores de clasificación del dataset con partición aleatoria.

Falsos negativos	**Falsos positivos**
(alula, pluma)	(lecho, globo)
(búfer, dispositivo de memoria)	(himeneo, amabilidad)
(solicitud de crédito, solicitud)	(champaign, alimentación)
(atmósfera, gas)	(kuusiku, juez)
(modernidad, actualidad)	(pintor, nivel)
(mente, recuerdo)	(gota, cuantía)
(campán, vino espumoso)	(proyector, financiamiento)
(trueno, ruido)	(pandereta, sentido)
(placer, estimulo positivo)	(surco, tuapse)

Cuadro 5.4: Ejemplos de errores de clasificación del modelo de concatenación en el dataset con partición aleatoria.

Sobre las instancias de falsos negativos no se detectaron particularidades. Pero en las instancias de falsos positivos se presentan casos interesantes como el de (himeneo, amabilidad) y (gota, cuantía), que si bien no denotan claramente una relación de hiperonimia, puede apreciarse cierto vínculo semántico.

Verdaderos positivos	**Verdaderos negativos**
(dipsómano, borracho)	(fiestero, guarderías)
(brzostek, pueblo)	(oscilador, rioseco)
(contradicción, creencia)	(baloncesto, amosis)
(humanoide, robot)	(violoncello, longevidad)
(sufrimiento, experiencia)	(demonio, bundestag)
(prohibición, etapa)	(cafeteros, nido)
(bacante, amigo)	(chamanes, club de atletismo)
(inicio, principio)	(feriantes, dobletes)
(ataque, golpe)	(autor, andadura)

Cuadro 5.5: Ejemplos de instancias bien clasificadas del modelo de concatenación en la partición aleatoria.

Dentro de las instancias de verdaderos negativos no se han encontrado casos que presenten una relación semántica entre las palabras. Sobre las instancias de verdaderos positivos es interesante comentar que se observaron casos de sinonimia donde la relación de hiperonimia se cumplen en ambos sentidos. Por ejemplo, casos como: (inicio, principio) y (ataque, golpe).

Modelo de order embedding

El modelo basado en el order embedding sobre la partición aleatoria se obtuvieron 14 (1,0 %) falsos negativos y 631 (15,8 %) falsos positivos. Y sobre la partición léxica se obtuvieron 10 (1,9 %) falsos negativos y 488 (31,7 %) falsos positivos. A continuación se muestran algunos de estos errores de clasificación.

Falsos negativos	**Falsos positivos**
(sinopia, ocre)	(colesterol, mitzvah)
(leprechaun, gremlin)	(multitud, dipolo eléctrico)
(heulandita, zeolita)	(talla, catarro)
(bison bonasus, bison)	(protein, poder)
(fachada, fraude)	(horror, aislador)
(toque de bocinas, ruido)	(urbe, granja)
(scratch, ruido)	(cautela, suspense)
(coelophysis, ceratosaurus)	(piano, perspectiva)
(longevidad, vejez)	(maestría, otorgamiento)

Cuadro 5.6: Ejemplos de errores de clasificación del modelo de order embedding en el dataset con partición léxica.

Todas las instancias de los falsos negativos son de relación de hiperonimia y son los errores del aprendizaje del modelo. Pero dentro de los falsos positivos aparecen pares como (cautela, suspense), (piano, perspectiva) y (maestría, otorgamiento), los cuales sí cumplen la relación de hiperonimia en determinados contextos. Estos casos son interesantes ya que aunque en el dataset estaba indicado como caso negativo por error, el modelo logró clasificar efectivamente como la instancia positiva.

Verdaderos positivos	**Verdaderos negativos**
(corazón, punto medio)	(historia, integrismo)
(fraude, crimen)	(apellido, pilote)
(viajero, persona)	(inocente, néstor)
(mejora, diferencia)	(rancheros, sucesores)
(apuro, dificultad)	(señoras, arca)
(enamorado, amigo)	(sombrilla, contubernio)
(veterano, viejo)	(pizarra, taiwanesas)
(cordero, carne)	(degüello, chelo)
(mente, conocimiento)	(temor, voluntarismo)

Cuadro 5.7: Ejemplos de instancias bien clasificadas del modelo de order embedding en la partición léxica.

Sobre las instancias de verdaderos negativos no se observan particularidades. En las instancias de verdaderos positivos aparecen pares de relación

sinónima como: (veterano, viejo) y (mente, conocimiento). También aparecen los pares donde la relación de hiperonimia se aplica sobre cierto contexto determinado como los casos de (mejora, diferencia), (apuro, dificultad) y (enamorado, amigo).

Capítulo 6

Conclusión y trabajo futuro

En el presente trabajo mostramos los resultados obtenidos en la detección supervisada de hiperonimia en español. Dada la falta de recursos en español para la detección de hiperonimia, creamos un conjunto de datos basado en trabajos previos para el inglés. Incluimos dos versiones del conjunto de datos según los subconjuntos para el entrenamiento, validación y evaluación, y la intersección léxica entre ellos, denominados: partición aleatoria y partición léxica. La primera se realiza al azar, mientras que la partición léxica no contiene intersección léxica entre las particiones, abordando el problema de memorización léxica en la detección de hiperonimia.

Entrenamos diferentes modelos neuronales utilizando vectores de palabras de propósito general entrenados con fastText y mostramos los resultados obtenidos. Mostramos el comportamiento de incluir cohipónimos y pares extraídos con la utilización de patrones durante el entrenamiento sin mejoras considerables. Este trabajo fue aceptado en el workshop *Linked Data in Linguistics* (LDL-2020) de LREC 2020.

Se mencionan como posibles líneas de trabajo futuro:

- Enriquecer el dataset con vocabularios de áreas específicas, ya sea relacionado a la terminología médica, industrial, etc.
- Abordar el problema de los múltiples significados de las palabras y los errores que introduce en la hiperonimia.
- Utilizar recursos existentes en otros idiomas para ampliar y/o mejorar

el dataset en español.

- Extender el trabajo a otras relaciones léxicas.

Bibliografía

Elena Akhmatova. Using hypernymy acquisition to tackle (part of) textual entailment. 01 2009. doi: 10.3115/1708141.1708152.

Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. Dbpedia: A nucleus for a web of open data. volume 6, pages 722–735, 01 2007. doi: 10.1007/978-3-540-76298-0_52.

C.I.B. Aura Josefina Rios Rios. *Razonamiento verbal y pensamiento analogico*. Colección Lecciones Escuela de Medicina y Cienciasde la Salud. Universidad del Rosario, 2009. ISBN 9789587380835. URL `https://books.google.com.uy/books?id=xbGxbz9uO7kC`.

Alfonso Ballesteros. Desarrollo de un marco de trabajo para el diseño de redes neuronales en java: Jredesneuronales, 2006. URL `http://www.redes-neuronales.com.es/`.

Marco Baroni, Raffaella Bernardi, Ngoc-Quynh Do, and Chung-chieh Shan. Entailment above the word level in distributional semantics. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, EACL '12, pages 23–32, Stroudsburg, PA, USA, 2012. Association for Computational Linguistics. ISBN 978-1-937284-19-0. URL `http://dl.acm.org/citation.cfm?id=2380816.2380822`.

Marco Baroni, Georgiana Dinu, and Germán Kruszewski. Don't count, predict! a systematic comparison of context-counting vs. context-predicting semantic vectors. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 238–247, Baltimore, Maryland, June 2014. Association for Computational

Linguistics. doi: 10.3115/v1/P14-1023. URL https://www.aclweb.org/anthology/P14-1023.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. A neural probabilistic language model. *J. Mach. Learn. Res.*, 3:1137–1155, March 2003. ISSN 1532-4435. URL http://dl.acm.org/citation.cfm?id=944919.944966.

James Bergstra and Yoshua Bengio. Random search for hyper-parameter optimization. *Journal of machine learning research*, 13(Feb):281–305, 2012.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146, 2017. doi: 10.1162/tacl_a_00051. URL https://www.aclweb.org/anthology/Q17-1010.

Francis Bond and Ryan Foster. Linking and extending an open multilingual Wordnet. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1352–1362, Sofia, Bulgaria, August 2013. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/P13-1133.

Cristian Cardellino. Spanish Billion Words Corpus and Embeddings, August 2019. URL https://crscardellino.github.io/SBWCE/.

IBM Knowledge Center. Neuronal network. URL https://www.ibm.com/support/knowledgecenter/es/SS3RA7_15.0.0/com.ibm.spss.modeler.help/idh_neuralnet.htm.

François Chollet. Keras: The python deep learning library. URL https://keras.io/.

Fabián Coelho. Significado de sinónimo, 2019. URL https://www.significados.com/sinonimo/.

Sarthak Dash, Md Faisal Mahbub Chowdhury, Alfio Gliozzo, Nandana Mihindukulasooriya, and Nicolas Rodolfo Fauceglia. Hypernym detection using strict partial order networks, 2019.

J. Daudé, Lluís Padró, and Gemma Oliver. Making wordnet mappings robust. *Procesamiento del lenguaje natural, ISSN 1135-5948, Nº. 31, 2003, pags. 47-54*, 01 2003.

Claudia Denicia-Carral, Manuel Montes-y Gómez, Luis Villaseñor-Pineda, and René García Hernández. A text mining approach for definition question answering. In Tapio Salakoski, Filip Ginter, Sampo Pyysalo, and Tapio Pahikkala, editors, *Advances in Natural Language Processing*, pages 76–86, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg. ISBN 978-3-540-37336-0.

S. T. Dumais, G. W. Furnas, T. K. Landauer, S. Deerwester, and R. Harshman. Using latent semantic analysis to improve access to textual information. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '88, pages 281–285, New York, NY, USA, 1988. ACM. ISBN 0-201-14237-6. doi: 10.1145/57167.57214. URL `http://doi.acm.org/10.1145/57167.57214`.

Christiane Fellbaum. *A Semantic Network of English: The Mother of All WordNets*, pages 209–220. 03 1998. doi: 10.1023/A:1001181927857.

Yansong Feng and Mirella Lapata. Visual information in semantic representation. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 91–99, Los Angeles, California, June 2010. Association for Computational Linguistics. URL `https://www.aclweb.org/anthology/N10-1011`.

Ruiji Fu, Jiang Guo, Bing Qin, Wanxiang Che, Haifeng Wang, and Ting Liu. Learning semantic hierarchies via word embeddings. In *ACL (1)*, pages 1199–1209. The Association for Computer Linguistics, 2014a.

Ruiji Fu, Jiang Guo, Bing Qin, Wanxiang Che, Haifeng Wang, and Ting Liu. Learning semantic hierarchies via word embeddings. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1199–1209, Baltimore, Maryland, June 2014b. Association for Computational Linguistics. doi: 10.3115/v1/P14-1113. URL `https://www.aclweb.org/anthology/P14-1113`.

Aitor Gonzalez-Agirre, Egoitz Laparra, and German Rigau. Multilingual central repository version 3.0. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC'12)*, pages 2525–2529, Istanbul, Turkey, May 2012. European Language Resources

Association (ELRA). URL http://www.lrec-conf.org/proceedings/lrec2012/pdf/293_Paper.pdf.

Google. An end-to-end open source machine learning platform. URL https://www.tensorflow.org/.

Edouard Grave, Piotr Bojanowski, Prakhar Gupta, Armand Joulin, and Tomas Mikolov. Learning word vectors for 157 languages, 2018.

Zellig S. Harris. Distributional structure. pages 146–162, 1954. doi: 10.1080/00437956.1954.11659520.

Marti A. Hearst. Automatic acquisition of hyponyms from large text corpora. In *14th International Conference on Computational Linguistics, COLING 1992, Nantes, France, August 23-28, 1992*, pages 539–545, 1992. URL http://aclweb.org/anthology/C92-2082.

Geoffrey E. Hinton. Learning distributed representations of concepts. 12 1986.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9:1735–80, 12 1997. doi: 10.1162/neco.1997.9.8.1735.

Sergio Jimenez and George Dueñas. *LAR-WordNet: A Machine-Translated, Pan-Hispanic and Regional WordNet for Spanish: 16th Ibero-American Conference on AI, Trujillo, Peru, November 13-16, 2018, Proceedings*, pages 392–403. 11 2018. ISBN 978-3-030-03927-1. doi: 10.1007/978-3-030-03928-8_32.

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. Bag of tricks for efficient text classification, 2016.

Gun Woo Lee, Mathias Etcheverry, Daniel Fernandez, and Dina Wonsever. Supervised hypernym detection in spanish through order embeddings. 03 2020.

Omer Levy, Steffen Remus, Chris Biemann, and Ido Dagan. Do supervised distributional methods really learn lexical inference relations? In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 970–976, Denver, Colorado, May–June 2015. Association

for Computational Linguistics. doi: 10.3115/v1/N15-1098. URL https://www.aclweb.org/anthology/N15-1098.

Jiaqing Liang, Yi Zhang, Yanghua Xiao, Haixun Wang, Wei Yang Wang, and Pinpin Zhu. On the transitivity of hypernym-hyponym relations in data-driven lexical taxonomies. In *AAAI*, 2017.

Alessandro Lopopolo and Emiel van Miltenburg. Sound-based distributional models. In *Proceedings of the 11th International Conference on Computational Semantics*, pages 70–75, London, UK, April 2015. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/W15-0110.

Kevin Lund, Curt Burgess, and Ruth Atchley. Semantic and associative priming in high-dimensional semantic space. pages 660–665, 01 1995.

Paul McNamee, Rion Snow, Patrick Schone, and James Mayfield. Learning named entity hyponyms for question answering. In *Proceedings of the Third International Joint Conference on Natural Language Processing: Volume-II*, 2008. URL https://www.aclweb.org/anthology/I08-2112.

Microsoft. The microsoft cognitive toolkit. URL https://docs.microsoft.com/es-es/cognitive-toolkit/.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space, 2013.

MILA. Theano. URL http://www.deeplearning.net/software/theano/.

George Miller, R. Beckwith, Christiane Fellbaum, Derek Gross, and Katherine Miller. Introduction to wordnet: An on-line lexical database*. 3, 01 1991. doi: 10.1093/ijl/3.4.235.

Tom M. Mitchell. *Machine Learning*. McGraw-Hill, New York, 1997. ISBN 978-0-07-042807-2.

Ana María Fernández Montraveta. La construcción del wordnet 3.0 en español. pages 201–220, 01 2010.

Chigozie Nwankpa, Winifred Ijomah, Anthony Gachagan, and Stephen Marshall. Activation functions: Comparison of trends in practice and research for deep learning, 2018.

Rosa María Ortega, Cesar Aguilar, Luis Villaseñor, Manuel Montes, and Gerardo Sierra. Hacia la identificación de relaciones de hiponimia/hiperonimia en Internet. *Revista signos*, 44:68 – 84, 03 2011. ISSN 0718-0934. URL https://scielo.conicyt.cl/scielo.php?script=sci_arttext&pid=S0718-09342011000100006&nrm=iso.

Philipp Cimiano Paul Buitelaar and Bernado Magnini. Ontology learning from text: Methods, evaluation and applications. page 180, 06 2005.

Dabal Pedamonti. Comparison of non-linear activation functions for deep neural networks on mnist classification task, 2018.

Jeffrey Pennington, Richard Socher, and Christopher Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar, October 2014. Association for Computational Linguistics. doi: 10.3115/v1/D14-1162. URL https://www.aclweb.org/anthology/D14-1162.

Emanuele Pianta, Luisa Bentivogli, and C. Girardi. Multiwordnet: Developing an aligned multilingual database. 01 2002.

Lutz Prechelt. Early stopping - but when? 03 2000. doi: 10.1007/3-540-49430-8_3.

Reinhard Rapp. Word sense discovery based on sense descriptor dissimilarity. 12 2003.

Salvador Climent Roca. Individuación e información parte-todo; representación para el procesamiento computacional del lenguaje). 8, 2000.

Stephen Roller, Katrin Erk, and Gemma Boleda. Inclusive yet selective: Supervised distributional hypernymy detection. In *Proceedings of the 25th International Conference on Computational Linguistics (COLING 2014)*, pages 1025–1036, Dublin, Ireland, August 2014. URL http://www.cs.utexas.edu/users/ai-lab/pub-view.php?PubID=127459.

Enrico Santus, Alessandro Lenci, Qin Lu, and Sabine Schulte im Walde. Chasing hypernyms in vector spaces with entropy. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics, volume 2: Short Papers*, pages 38–42, Gothenburg, Sweden,

April 2014. Association for Computational Linguistics. doi: 10.3115/v1/E14-4008. URL https://www.aclweb.org/anthology/E14-4008.

Vered Shwartz, Yoav Goldberg, and Ido Dagan. Improving hypernymy detection with an integrated path-based and distributional method. *CoRR*, abs/1603.06076, 2016. URL http://arxiv.org/abs/1603.06076.

Gerardo Sierra, Rodrigo Alarcón, César Aguilar, and Carme Bach. Definitional verbal patterns for semantic relation extraction. *Terminology*, 14(1): 74–98, 8 2008. ISSN 0929-9971. doi: 10.1075/term.14.1.05sie.

Rion Snow, Daniel Jurafsky, and Andrew Y. Ng. Learning syntactic patterns for automatic hypernym discovery. In L. K. Saul, Y. Weiss, and L. Bottou, editors, *Advances in Neural Information Processing Systems 17*, pages 1297–1304. MIT Press, 2005. URL http://papers.nips.cc/paper/2659-learning-syntactic-patterns-for-automatic-hypernym-discovery.pdf.

Rion Snow, Daniel Jurafsky, and Andrew Ng. Semantic taxonomy induction from heterogenous evidence. 01 2006. doi: 10.3115/1220175.1220276.

Fabian M. Suchanek, Gjergji Kasneci, and Gerhard Weikum. Yago: A core of semantic knowledge. In *Proceedings of the 16th International Conference on World Wide Web*, WWW '07, page 697–706, New York, NY, USA, 2007. Association for Computing Machinery. ISBN 9781595936547. doi: 10.1145/1242572.1242667. URL https://doi.org/10.1145/1242572.1242667.

Kai Chen Tomas Mikolov, Ilya Sutskever, Gregory S Corrado, and Jeffrey Dean. Distributed representations of words and phrases and their compositionality. page 3111–3119, 2013.

P. D. Turney and P. Pantel. From frequency to meaning: Vector space models of semantics. *Journal of Artificial Intelligence Research*, 37:141–188, Feb 2010. ISSN 1076-9757. doi: 10.1613/jair.2934. URL http://dx.doi.org/10.1613/jair.2934.

Dmitry Ustalov, Nikolay Arefyev, Chris Biemann, and Alexander Panchenko. Negative sampling improves hypernymy extraction based on projection learning. 04 2017. doi: 10.18653/v1/E17-2087.

Ivan Vendrov, Ryan Kiros, Sanja Fidler, and Raquel Urtasun. Order-embeddings of images and language. *CoRR*, abs/1511.06361, 2015. URL http://arxiv.org/abs/1511.06361.

Piek Vossen. *Introduction to EuroWordNet*, pages 1–17. Springer Netherlands, Dordrecht, 1998. ISBN 978-94-017-1491-4. doi: 10.1007/978-94-017-1491-4_1. URL https://doi.org/10.1007/978-94-017-1491-4_1.

Denny Vrandecic. Wikidata: a new platform for collaborative data collection. In *WWW*, 2012.

Chengyu Wang, Xiaofeng He, and Aoying Zhou. Improving hypernymy prediction via taxonomy enhanced adversarial learning. In *Proceedings of the 33rd AAAI Conference on Artificial Intelligence*, 2019.

Julie Weeds, David Weir, and Diana McCarthy. Characterising measures of lexical distributional similarity. In *Proceedings of the 20th International Conference on Computational Linguistics*, COLING '04, Stroudsburg, PA, USA, 2004. Association for Computational Linguistics. doi: 10.3115/1220355.1220501. URL https://doi.org/10.3115/1220355.1220501.

Julie Weeds, Daoud Clarke, Jeremy Reffin, David J. Weir, and Bill Keller. Learning to distinguish hypernyms and co-hyponyms. In *COLING*, pages 2249–2259. ACL, 2014.