Facultad De Ingeniería UDELAR

Proyecto de grado

Detección de Hiperonimia con Word Embeddings en Español

Tutor:
Mathias, Etcheverry
Dina , Wonsever

Estudiante:
Gun Woo, Lee

Marzo, 2020

Índice

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 otras de significado más específico, pudiendo generalmente sustituir la unidad léxica más general por las más específicas o viceversa, dependiendo de los escenarios[1]. La identificación exitosa de esta relación mejora sustancialmente las tareas de PLN, como la respuestas a preguntas, extracción de información, sistemas de traducción automática, implicación textual y sistemas de búsqueda semántica.

Además, dentro del campo de la Ingeniería Lingüística, las relaciones de hiperonimia se han usado en tres áreas específicas:

- 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 la de Denicia [DMVH06], o la de Sierra [SAAB08].
- En el diseño de sistemas para la detección de unidades textuales cuya información léxica aporta un conocimiento determinado y ayuda a eliminar ambigüedad de una palabra en un contexto dado. Por ejemplos los trabajos de Hearst [Hea92] y de Snow [SJN06] se ubican en esta área.
- En el desarrollo de taxonomías y ontologías, cuya estructuración se basa en las relaciones de hiperonimia. Como los trabajos representativos se encuentran la de Snow [SJN06] y la de Buitelaar [PM05].

[1]Por Ej. No es válida la sustitución de “animales” por “perros” en la siguiente oración “Hay animales que corren, otros que vuelan y otros que nadan.”

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, también llamados **dataset**, de 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 relación de hiperonimia se ha abordado ampliamente en el idioma inglés, no hay tantos trabajos realizados en español. En especial según nuestro conocimiento hay escaso conjunto de datos disponibles en español para la detección de hiperonimia basado en método supervisado, por lo que creemos que el mismo proyecto es un aporte valioso para la disciplina de PLN en español. El aporte de éste 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. Para la extracción de dataset 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 correspondientes al dataset de entrenamiento construido anteriormente.

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 trata en el informe. En el capítulo 3 se presentan diferentes metodologías que se emplearon para la construcción del dataset en español y sus resultados correspondientes. En el capítulo 4 se trata de los modelos de redes neuronales construidos con sus detalles de implementación. Los resultados finales de los modelos se muestra en el capítulo 5. Por último en el capítulo 6 se indica el trabajo futuro y la conclusión de presente trabajo.

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 hiponimia-hiperonimia

Según la norma UNE (Asociación Española de Normalización), la relación hiponimia-hiperonimia (abreviado como hiperonimia) se define como *el concepto específico posee todas las características del genérico y, además como mínimo, una característica suplementaria distintiva*. Por ejemplo el concepto específico tulipán comparte todas las características del concepto genérico flor y además, como mínimo, una característica suplementaria distintiva que lo diferencia tanto de su hiperónimo flor como de sus cohipónimos rosa y geranio como es, por ejemplo, el hecho de que pertenezca a la familia de las Liliáceas.

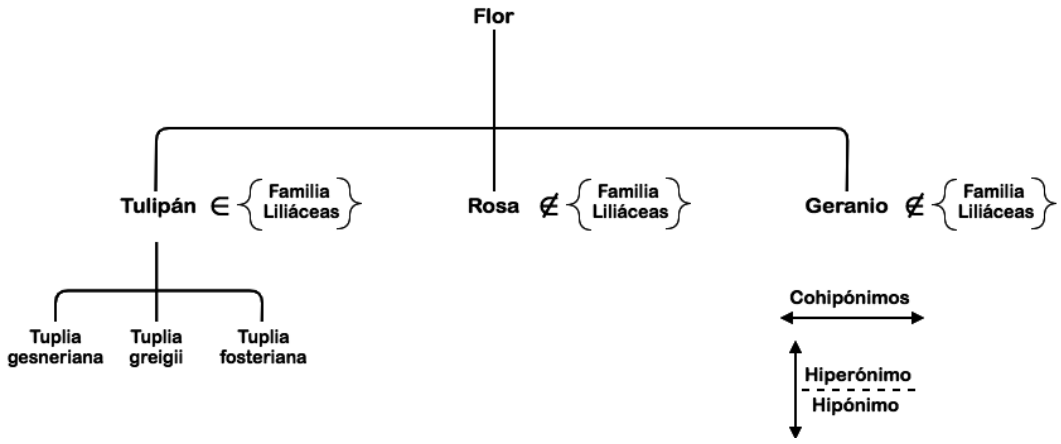


Figura 1: Relación de Hiperonimia

Tulipán, rosa y geranio son hipónimos de flor y cohipónimos entre ellos, y mantienen entre sí una relación de exclusión o incompatibilidad. 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. Asimismo, también cabe destacar que el hecho de designar un concepto como hipónimo o hiperónimo es relativo, no absoluto, ya que dependiendo de contexto puede

cumplir o no la relación de hiperonimia. Ver la Figura 1.

Miller (1993) 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 tipo de Y" [MBFGM91]. 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. Existen otras propiedades las cuales se cumplen en algunas ocasiones. Por ejemplo, puede haber hiperónimos múltiples, la relación hiperonimia puede variar de un idioma a otro, un mismo concepto puede aparecer en distintos lugares de la jerarquía o puede haber algún grupo de conceptos que se encuentran a un mismo nivel y comparten una serie de características, pero carecen de hiperónimo [RA07].

2.2. WordNet

WordNet es una gran base de datos lexical 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. WordNet se asemeja a un tesauro donde se ordena las palabras o los términos en forma de listado agrupando las palabras para representar sus conceptos. Pero a diferencia de un tesauro en WordNet en primer lugar las relaciones se establecen en base el sentido específico de las palabras. Es decir si dos palabras se encuentran muy próximos entre sí en la red se desambigan semánticamente. Además en WordNet cada relaciones semánticas entre las palabras están etiquetadas, mientras que en las agrupaciones de palabras en un tesauro no siguen ningún patrón explícito que no sea similitud de significado [Roc00].

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[2]. En los últimos años han sido creados WordNets individuales de diferentes idiomas los cuales enriquecieron la información léxica de otros idiomas, incrementando su uso general y aplicación en diverso 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 Procesamiento de Lenguajes Naturales (PLN) y Lingüística computacional. La última versión hecha pública es WordNet 3.1 la cual consta de 117.200 entradas entre palabras y grupo lexicalizados pertenecientes a las llamadas 'categorías abiertas': Nombres (69 %), Adjetivos (16 %), Verbos (12 %) y Adverbios (3 %).

2.2.1. Synset

Como se mencionó anteriormente la unidad básica en la que se estructura WordNet es el synset. Un synset o traducido como el conjunto de sinónimos, es una agrupación de palabras que son sustituibles en al menos un contexto obteniéndose un significado semejante, como entre las palabras caminar-andar o automóvil-coche. En otra palabra, en WordNet cada palabra esta asociada con uno o mas synsets que representa un mismo concepto lexicalizado. 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 distintos entonces la misma se representan en distintos synsets. Por ejemplo la palabra polisémica 'mouse', traducido como el ratón esta asociada a dos synsets diferentes como se muestra en el Cuadro 1. Por lo tanto, cada par de forma-significado en WordNet es único. Según Miller el significado en WordNet no es, como en otros planteamientos semánticos, composicional o construido, sino diferencial: el significado de un concepto viene dado por contraposición al del resto de conceptos de la base de datos.

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

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 1: Synsets de palabra 'mouse'

2.2.2. Relaciones léxicas

Así, en WordNet, las relaciones se establecen fundamentalmente entre conceptos semánticos o 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. 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,...). 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 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' [Coe]. Y WordNet se enfoca más en la sinonimia en contexto (parcial y contextual) como solución pragmática y realista que siempre sujeta a matices e interpre-

taciones, permitiendo afrontar la tarea de representar y tratar computacionalmente el conocimiento léxico-semántico de una lengua [Roc00].

- Hiperonimia
  A diferencia de la sinonimia, la relación de hiperonimia se establece a nivel semántico. La hiperonimia también llamados como subordinación-supraordinación, subconjunto-superconjunto, o la relación "ISA", vincula los synsets que representa un concepto lexicalizado mediante uso de los punteros. Dado que la hiperonimia es transitiva y asimétrica y que normalmente hay un solo superordinado es posible construir una estructura semántica jerárquica, donde los hipónimos son los mismos superordinados de los hiperónimos. Ver la Figura 2. Los hiperónimos hereda las característica del concepto más genérico y agrega al menos una característica que distingue de su superordinado. 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. Las definiciones de los sustantivos se construyen en base a las definiciones del término superordinado agregando los rasgos distintivos que diferencian de su superordinado [MBFGM91].

- Meronimia
  Otra relación importante en los synsets sustantivo es **meronimia**. 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. Ésta relación es representado en tres tipos: 'part-of' *(parte de)*, 'member-of' *(miembro de)* y 'substance-of' *(sustancia de)*. La primera de ellas relaciona a una entidad con sus componentes 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. En la Figura 2 se puede observar una representación jerárquica de los sustantivos según la relación de hiperonimia y de meronimia [Roc00].

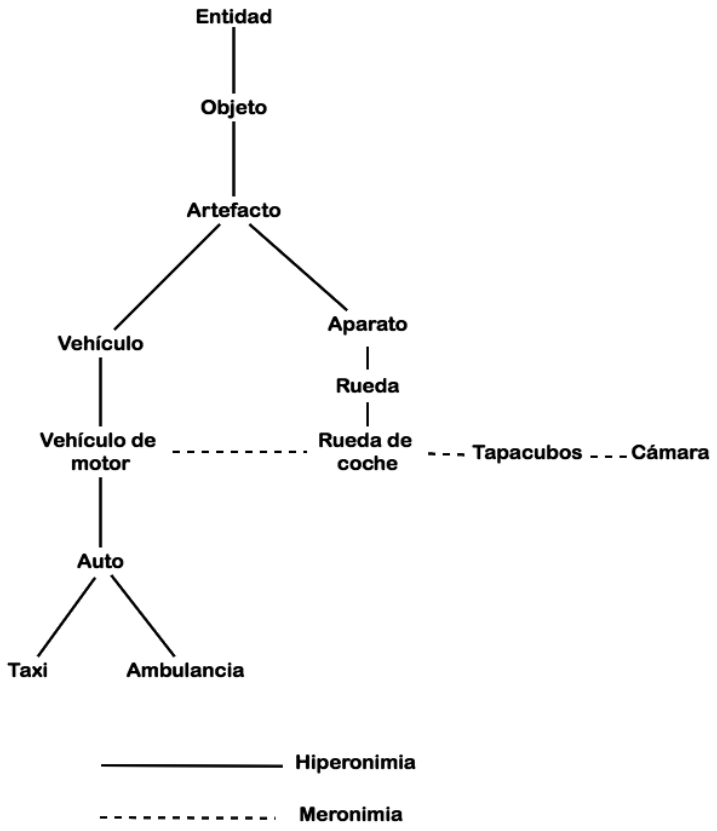


Figura 2: Relaciones léxicas[3]

- Antonimia y Similitud
  Según ANEP[4], la antonimia es una relación que se establece entre dos palabras cuyo significado es incompatible en un mismo contexto. Así, 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 la antonimia se establece entre las palabras léxicas y no entre los conceptos. Por ejemplo, los significados {subir, ascender} y {caer, descender} pueden ser opuestos conceptualmente, pero no son antónimos, mientras subida-caída y ascender-

[3]Como se mencionó anteriormente las relaciones se establecen entre los synsets y no entre las palabras. Cada palabras que aparecen en la figura representa un synset como Synset('entidad'), Synset('Auto'), etc

[4]Administración Nacional de Educación Pública de Uruguay

descender se consideran como antónimos. Si bien se presenta la relación de antonimia entre los sustantivos, no se considera como una relación fundamental para la organización de los mismos. Pero sí es considerado como un principal relación para la organización de los adjetivos y adverbios.

En cuanto los adjetivos, WordNet divide en dos clases principales: descriptivos y relacionales. 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 formados por un par de adjetivos que cumple 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". Y 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 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. Ver la Figura 3.

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 adjetivo sólo modifica la referencia y no al referente. Por ejemplos, "El anterior presidente" o "Mi viejo amigo" [MBFGM91].

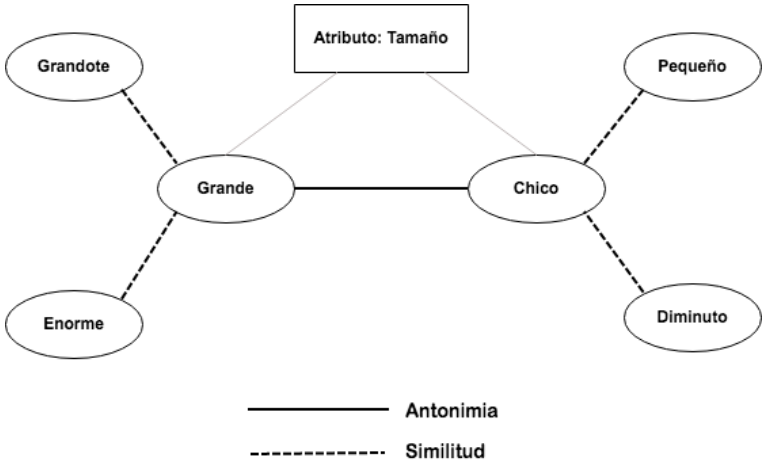


Figura 3: Adjetivos en Wordnet

- 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.

  Las diferentes relaciones que vinculan los verbos se organizan en base consecuencia lógica (Implicación - Entailment). Ésta 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'. Lo contrario de vinculación es la contradicción: si la oración que él está roncando implica que está durmiendo, entonces está roncando también contradice la frase que no está durmiendo.

- 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 sean coextensivos temporalmente. 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 son coextensivo. En contraste con pares como roncar y dormir, o comprar y pagar, están relacionados solo por implicación e inclusión temporal apropiada. Y estas actividades no pueden coexistir. 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 4.

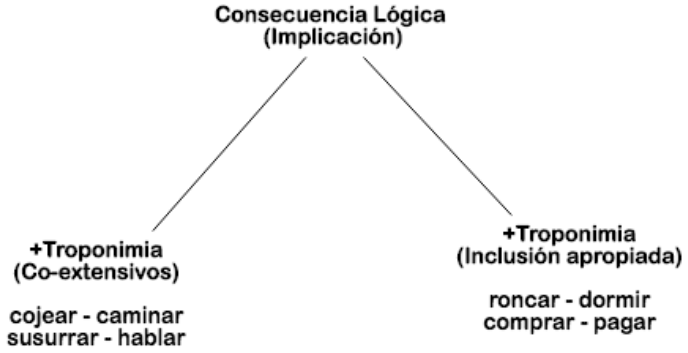


Figura 4: Relación de Troponimia

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 [MBFGM91].

En cuanto a los adverbios, sólo hay pocos adverbios en WordNet ya que la mayoría de los adverbios en inglés se derivan directamente de los adjetivos a través de la fijación morfológica. Por ejemplo, 'sorprendentemente', 'extrañamente', etc. 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.

2.3. WordNet en otros idiomas

A pesar de que WordNet fue creado para el idioma inglés, durante los últimos años han sido creados WordNets individuales de diferentes idiomas. Diversos proyectos han desarrollado WordNets 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[5] se puede ver una lista exhaustiva de los WordNets disponibles para diversas lenguas.

Existen dos estrategias principales para construir WordNet para lenguas distintas del inglés [Vos98].

- **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 entre

[5] http://globalwordnet.org/

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 WordNets. La desventaja es que estos WordNets son demasiado influenciados por PWN y pueden arrastrar errores y contener algunas deficiencias estructurales.

Dentro de WordNets que fueron construidos en base la estrategia de expansión y tienen alto nivel de cubrimiento en comparación con PWN se encuentran: MultiWordNet [PBG02], Multilingual Central Repository [GLR12], WNE [Mon10], LAS-WordNet [JD18].

2.3.1. 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 se incluye o licencia. Open Multilingual Wordnet (OMW) fue construido para acceder a múltiple WordNet a través de una única interfaz común, además los datos son semi-estructurados que tienen mínima restricciones para permitir extender WordNet existentes de forma gratuita y crear WordNet adicionales que pueden ser agregado a la misma red de OMW abierta.

La construcción de OMW fue basado en la estrategia de expansión y posteriormente lo mismo se extendió con otro fuente de dato diferente a WordNet como Wiktionary. Por ejemplo, dado synset $dog_{n:1}$ de PWN que representa el concepto del perro como animal; se le agrega **lemmas** (unidad que representa un sentido específico) de diferentes idiomas como *chien* en francés, *perro* en español. Especificamente en OMW el vínculo se establece entre los sentidos de los WordNets, manteniendo la jerarquía principal entre los synsets de PWN. Como los WordNets individuales se encuentran enlazados a diferentes versiones de PWN, mediante los scripts de mapeo creado por F. Bond, 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. F. Bond extiende OMW agregando datos externa a los WordNet individuales, combinando datos recopilados de forma automática de Wiktionary y Unicode Common Locale Data Repository

(CLDR) [DPO03]. Como resultado, logró construir Open WordNet multilingüe de alta calidad 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 la descarga y también es posible acceder y manipularlos mediante la herramienta NLTK [BF13].

2.4. Word embeddings

Word Embeddings es una técnica que mapea palabras a un vector n-dimensional de números reales. Dicha representación tiene propiedades de agrupamiento útiles, ya que agrupa palabras que son semánticamente y sintácticamente similares. Por ejemplo dadas las palabras "delfín", "foca" y "mesa". Las palabras "delfín" y "foca" deben encontrarse más cerca entre sí, que de la palabra "mesa", la cual no tiene una vinculación semánticamente fuerte con las otras dos. Las palabras se representan como vectores de valores reales, donde cada valor captura una dimensión del significado la palabra. Esto provoca que palabras semánticamente similares, tengan vectores similares. En otras palabras, cada dimensión de los vectores representa un significado y el valor numérico en cada dimensión captura la cercanía de la asociación de la palabra a dicho significado. En el Cuadro 2 se muestra una simplificación de esta idea donde cada columna representa el vector de una palabra y cada fila representa una dimensión de significado. Por ejemplo dada la dimensión de 'Masculinidad', el valor de significado de la palabra 'Rey' es mayor que el valor de la palabra 'Reina'. El objetivo de Word Embeddings es cuantificar y categorizar similitudes semánticas entre elementos lingüísticos. Es de esperar que sinónimos y palabras intercambiables se encuentren cerca en tal espacio vectorial.

	Rey	Reina	Mujer	Princesa	...
Realeza	0.99	0.99	0.02	0.98	...
Masculinidad	0.99	0.05	0.01	0.02	...
Feminidad	0.05	0.93	0.99	0.94	...
Edad	0.7	0.6	0.5	0.1	...
...	...	...	...	...	...

Cuadro 2: Ejemplo simplificado de vectores de palabra

En las últimas décadas los Word Embeddings se han convertido en una de las corrientes principales dentro del PLN. Pues los Word Embeddings capturan el significado de las palabras, y las traducen a una codificación que puede ser usada como entrada para todo tipo de redes neuronales. Esto ha provocado que su uso se haya extendido rápidamente y actualmente sean una pieza fundamental en la arquitectura de todo tipo de modelos que realizan tareas de PLN [Gar18].

2.4.1. Métodos de construcción de word embeddings

Actualmente existen numerosos métodos para construir los vectores de palabras. Dentro de los más conocidos se encuentran:

- **Word2Vec**: Se trata de un grupo de modelos predictivos de generación de word embeddings. Puede utilizar el modelo **Continuous Bag-of-Words (CBOW)**, así como también el modelo **Skip-gram**. En el primero, dado el contexto de la palabra objetivo, intenta predecirla. En el segundo, dada la palabra intenta predecir el contexto [MCCD13]. Estos modelos se presentan detalladamente en la sección 2.6.2.

- **FastText**: FastText es una extensión del modelo Word2Vec. Cada palabra es tratada como la suma de sus composiciones de caracteres llamados ngrams. El vector para una palabra está compuesto por la suma de sus ngrams. Por ejemplo el vector para la palabra "apple" está compuesto por la suma los vectores para los ngrams "<ap, app, appl, apple, apple, ppl, pple, ple, le>". La mayor ventaja es que permite obtener mejores representaciones para palabras "extrañas", las cuales cuentan con muy pocas apariciones en corpus de textos, y así poder generar vectores para palabras que no se encuentran en el vocabulario de los word embeddings [MGBPJ17].

- **GloVe**: A diferencia de los métodos anteriores, GLOVE genera una matriz de conteo donde se almacena la información de la concurrencia entre palabras y contextos. Cada elemento de la matriz representa la cantidad de veces que aparece una palabra en algún contexto. Luego a la matriz se realiza una factorización global aplicando técnicas como la descomposición en valores singulares, y de la matriz resultante se obtiene los vectores correspondientes de las palabras [PSM14].

2.5. 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 (equivalente a las máquinas de Turing) basado en el comportamiento de las neuronas [Bal].

2.5.1. 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

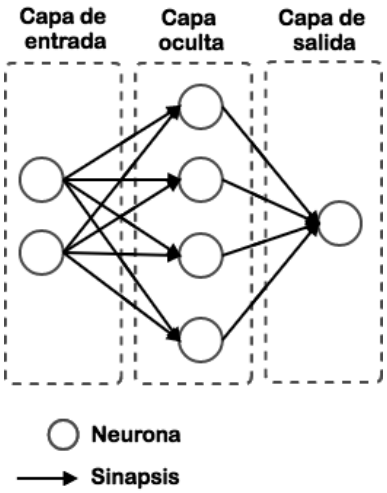


Figura 5: Esquema de una red neuronal simple

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 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 y la red sigue mejorando sus predicciones hasta haber alcanzado uno o varios criterios de parada.

Al principio, todas las ponderaciones son aleatorias y las respuestas que resultan de la red son, posiblemente, disparatadas. 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 [Cen].

2.5.2. Elementos de una RNA

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

- 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

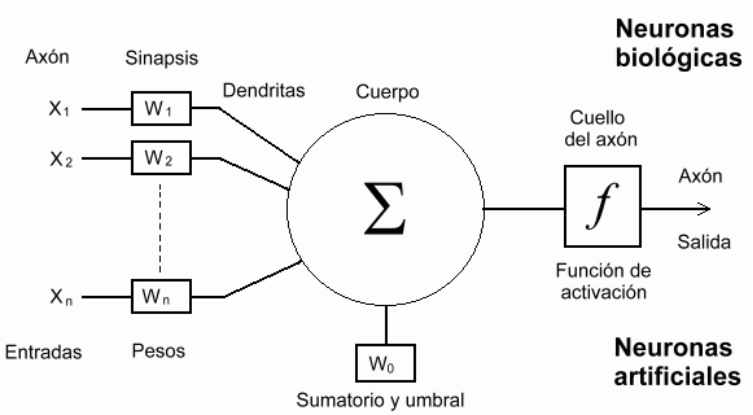
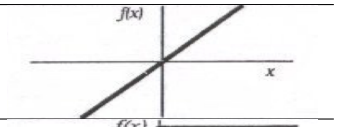
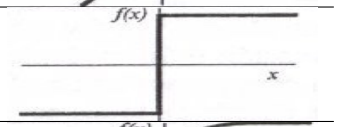
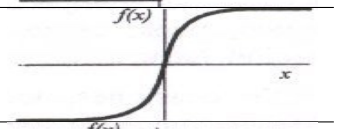
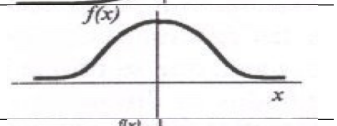
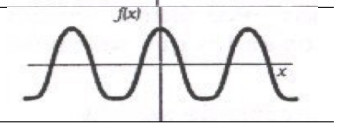


Figura 6: Elementos de una red neuronal

Cada neurona recibe n entradas x_i de la neurona anterior $i - esima$, ponderadas con los pesos sinápticos $w_i j$. 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 6. En este modelo, la salida neuronal Y está dada por:

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

La función de activación se elige de acuerdo a la tarea realizada por la neurona. Entre las más conocidos podemos destacar:

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

Cuadro 3: Función de activación

2.5.3. Métricas del modelo

Las métricas más usadas para evaluar el rendimiento de modelo supervisado en tareas de clasificación son: *accuracy* - exactitud, *precision* - precisión, *recall* - exhaustividad y f1. 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 4, donde $\_N$ / $\_P$ (*Negativo / Positivo*) representa la predicción del modelo y $T\_$ / $F\_$ (*True / False*) representa si la misma predicción fue correcta o no respectivamente.

Tendiendo en cuenta la matriz anterior, 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.

$$f1 = 2 \cdot \frac{precision \cdot recall}{precision + recall}$$

El valor *f1* 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 4: Matriz de confusión

2.5.4. Problemas adecuados para utilizar una RNA

Cualquier problemas que contemplan los siguientes aspectos pueden ser resueltos por una RNA:

- Instancias son representadas por vectores atributo-valor.
- La salida de la función puede ser un valor o un vector, discreto o real.
- Los datos de entrenamiento pueden contener errores.
- Largos tiempos de entrenamiento son aceptables.
- Respuesta rápido de ejecución de la red para dar una salida.
- La necesidad de interpretar la función aprendida no es relevante; la red es una "caja negra".

2.6. 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 posteriormente se han desarrollado 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 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 representaciones distribucionales de los dos términos, es decir, los contextos con los que cada término aparece por separado en el corpus. Dentro del enfoque distribucional, es posible usar métodos supervisados como no supervisados. En métodos supevisados, el par de términos X e Y está representado por un vector de características utilizando la técnica de Word Embeddings (Sección 2.4).

2.6.1. Métodos basados en patrones-caminos

Este método fue propuesto primera vez por Hearst [Hea92]. Hearst 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 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 5: Patrones que indican la relación de hiperonimia

El método de Hearst se basa en patrones construidas manualmente y funciona para descubrir determinadas relaciones semánticas como la hiperonimia. Pero es un método tedioso y es limitado por el pequeño número de patrones empleados. Pues los patrones de Hearst no cubren todas las ocurrencias de la relación de hiperonimia.

Posteriormente Snow logró construir un clasificador automático más genérico para detectar relaciones semánticas además de la hiperonimia [SJN05]. 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 método tradicional, Snow utiliza el camino 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 sintáctico que representa las relaciones sintácticas entre las palabras con sus categorías gramaticales correspondientes. En otras palabras, en lugar de usar los patrones específicos cada par de términos X e Y, los representa como *el conjunto múltiple de todos los caminos de dependencia que conectan X e Y en el corpus*. Luego en base a estos caminos es entrenado el clasificador. Los caminos que indican la relación de hiperonimia son aquellos a los que el clasificador les asignó pesos más altos.

Los patrones identificados por éste método se presentan dentro de los patrones encontrados por Hearst y logró obtener mejor rendimiento. Además se

demostró que estos métodos automáticos pueden ser competitivos con WordNet para la identificación de pares de hiperonimia en corpus de Newswire [SJN05].

Asimismo existen trabajos que extraen pares de hiperonimia a partir de corpus en Español. Ortega logró identificar relación de hiperonimia a partir de una serie de patrones léxicos no sintáctico[6] en español [OAVMS11]. 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.

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 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 filtración se extrae 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 6.

[6]Ejemplo de un patrón léxico no sintáctico: los (palabras)* y otros (palabras)*.
Ejemplo de un patrón léxico-sintáctico: NP {NP,}*{,} y otros NP

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 6: Patrones de Ortega con su valor de confianza

2.6.2. Métodos distribucionales

El problema principal de los métodos basados en caminos-patrones es que las palabras deben ocurrir exactamente según el orden definido por patrones, de lo contrario no se puede detectar ninguna relación. Otra alternativa para detectar la relación de hiperonimia es basado en representación distribucional, donde las palabras son representados como vectores en función de su distribución en corpus de gran escala.

Este método se basa en las hipótesis de distribución. Harris fue el primero en plantear la idea sobre que existe una correlación entre la similitud de distribución[7] y la similitud semántica. Considera que el significado de una palabra está determinado por el contexto en qué se usa, y que importantes aspectos del significado de una palabra son una función o pueden estar representados por el conjunto de contextos en los que aparece. Las diferencias de significado están en correlación con las diferencias de distribución. Esto implica que el significado de una palabra se puede representar mediante la suma de contexto en qué aparece [Har54].

Esta idea permitió desarrollar los modelos de semántica distribucional, también conocidos como modelos de "espacio de palabras" o de "similitud de distribución". Estos modelos construyen representaciones semánticas de manera dinámica en forma de espacios vectoriales multidimensionales a través

[7]La distribución de un elemento A se define como la suma de todos sus entornos. Y cada entorno es una matriz de los elementos concurrentes de A, cada uno en su posición particular.

del análisis estadístico de los contextos en que cada palabra aparece. Una palabra es representado como un vector donde cada uno esta formado por los contextos en que ésta aparece y por el número de veces que ocurre en cada uno de ellos. 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 considera. Cada fila es un vector que tiene codificada numéricamente la concurrencia de la palabra que da lugar a la fila con todos los contextos posibles, que son las columnas.

Por ejemplo es posible formar una matriz de concurrencia M, donde cada fila representa un vector $X_{palabra}$ que describe el uso de la palabra en diferentes contextos del corpus.

	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 7: Matriz de concurrencia M

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 el Cuadro 7.

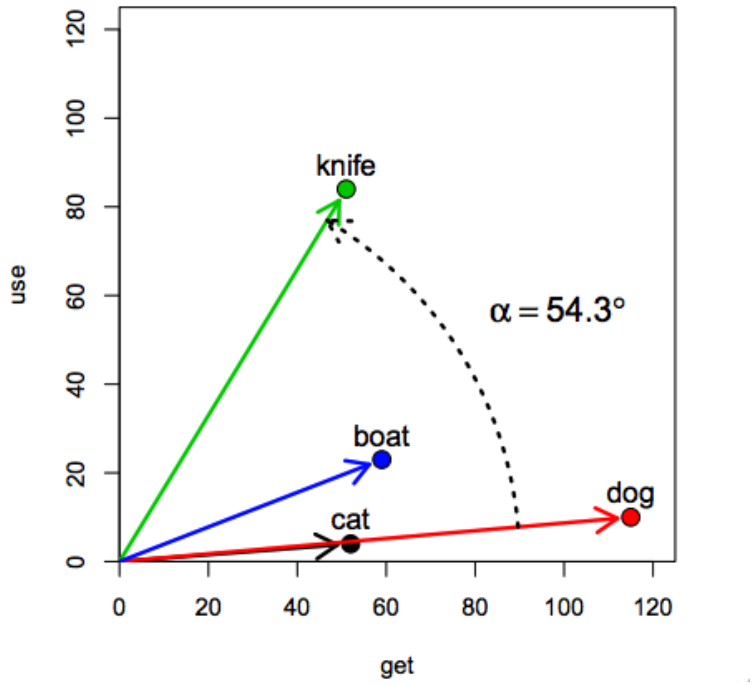


Figura 7: Matriz de concurrencia M

Esto permite aproximar de forma cuantitativa la semántica de una palabra en términos de representaciones geométricas 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 de manera objetiva, considerando 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 el 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 asemeja las semánticas de las dos palabras.

La implementación de estos modelos varían según el contexto y el mecanismo de abstracción utilizado. 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 [FL10] o audio [LM15]. En cuanto al mecanismo de abstracción utilizado existen los

métodos clásicos ya visto anteriormente que se basa en estadísticas de concurrencias entre palabras y contextos, lo cual se llama **basados en conteo**. Por otro lado se encuentra los métodos **basados en predicción**, que en su lugar aplican técnicas de aprendizaje automático para inducir representaciones basadas en un tarea de predicción.

El problema mayor de los métodos basados en conteo es cuando el corpus es muy grande, la construcción de matriz de concurrencia y el procesamiento de la matriz se convierten en una tarea arduo. Por lo general esto se divide en cuatro etapas [TP10]. Primero se construye la matriz de concurrencia en base un determinado elemento (palabra, par de palabra, término) y la cantidad de veces que ocurrió en una determinada situación (contexto, patrón, 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, es necesario suavizar la matriz para reducir la cantidad de ruido aleatorio mediante el método de descomposición en valores singulares [Rap03]. Por último calcular la similitud entre los vectores mediante algunas medidas de similitud como coseno, Jaccard, Jensen-Shannon, L1 [Wee04]. Por ejemplo dado x e y, la medida coseno mencionado anteriormente se calcula como:

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

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

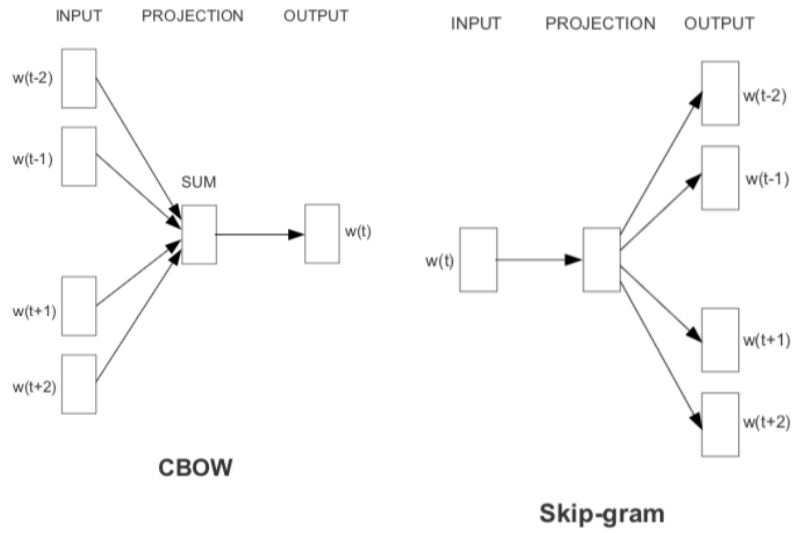
Los principales modelos mas conocidos en base este enfoque se puede mencionar el modelo LSA, también llamado como modelo de análisis latente semántico [DFLDH88], y el modelo HAL también conocido como análogo en el hiperespacio para el lenguaje [LBA95].

La desventaja mayor de éste método es que el proceso de optimización de los vectores no está supervisado y se basa en consideraciones independientes basado en teóricas de la información y cálculo matemático. Además esta aproximación cuantitativa de la semántica de una palabra se complica más cuando se trata de unidades lingüísticas complejas, más allá de las palabras. Es posible enmarcar el problema de estimación del vector directamente como una tarea supervisada, donde los pesos de los vectores se configuran directamente para predecir de manera óptima los contextos en los que las palabras correspondientes tienden a aparecer. A esta nueva alternativa se le llama modelos de predicción o métodos basados en predicción [BDK14]. Aquí los vectores no representa los datos estadísticos acerca de la distribución de una palabra en determinados contextos, sino los vectores son una representación simbólico multidimensional de una palabra para los modelos de redes neuronales.

La idea propuesta por Geoffrey fue usar los pesos de las capas internas de redes neuronales para representar un concepto particular en un espacio continuo [Hin86]. Y esta técnica aplicada al lenguaje natural junto con redes neuronales recursivas permitió resolver la tarea de predicción de oraciones utilizando los pesos internos aprendidos por la red como representación de las palabras. Luego Bengio toma esta idea y plantea una arquitectura que toma vectores continuos de palabras y una función paramétrica definida sobre dicho espacio de vectores, el cual modela la probabilidad de la siguiente palabra dada una ventana de N palabras precedentes [BDVJ03]. En el trabajo posterior realizado por Mikolov 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 8a), define una ventana de largo fijo y simétrica alrededor de una palabra central y busca optimizar la predicción de la palabra central 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 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 8: Modelos de Mikolov

Como resultado lograron construir vectores de palabras de alta calidad de semántica con gran eficiencia computacional. Los vectores resultantes de estos modelos pueden responder relaciones semánticas como una ciudad y el país al que pertenece por ejemplo Francia-Paris y Berlín-Alemania.

Los dos métodos anteriores descritos fueron desarrollados en base idea diferente y desarrollados de forma independiente donde tiene un objetivo común en buscar la relación léxica y estudiar la similitud entre las palabras. Existen diversos trabajos donde realizan comparaciones de los dos métodos por medio de los resultados obtenidos en cada uno de los métodos que pueden ser aplicados en diferentes escenas del problema de léxica semántica. Por ejemplo en el trabajo de Baroni considera diferentes conjuntos de pruebas como para los problemas de similitud entre palabras, detección de sinonimia, categorización del concepto y analogía. Y como resultado obtuvo mejor rendimiento los métodos basados en predicción en la mayoría de las pruebas realizadas. En algunas tareas específica se demostró que los métodos basados en conteo logran mejor rendimiento [BDK14].

No obstante, según Levy 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 positivos, puede aprender que la palabra *'animal'* sea un hiperónimo prototípico, clasificando cualquier tupla nuevo *(x, animal)* como positivo, independientemente de la relación semántica entre x y *animal*. 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 [LRBD15].

2.6.3. Métodos híbridos

En los recientes trabajos utilizan ambos enfoques basados en patrones-caminos y basados en distribución para abordar la tarea de detección de hiperonimia como el trabajo de Shwartz [SGD16]. En el mismo trabajo se utiliza un algoritmo híbrido, en el que caminos de dependencia se codifican utilizando una red neuronal recurrente, que logra resultados comparables a los métodos distribucionales. Luego, se amplía el enfoque integrando señales distribucionales, con lo cual se mejora significativamente el estado del arte en ésta tarea.

3. Construcción de dataset en español

El objetivo principal de esta parte es construir el primer prototipo de dataset. El dataset consiste en un conjunto de tuplas de tres elementos donde los primeros dos son palabras léxicos en español, y el último un booleano que indica 'True' o 'False' si entre las dos palabras se cumple la relación de hiperonimia o no respectivamente. Aquellas tuplas que contienen dos palabras que cumple la relación de hiperonimia, las llamamos **dataset positivo** y se anota de siguiente forma *(hipónimo, hiperónimo, True)*. Por ejemplo *(perro, animal, True)* y *(martillo, herramienta, True)*. Las demás tuplas que no cumplen la relación de hiperonimia son llamadas **dataset negativo**. En el informe para simplificar abreviamos la anotación de las tuplas con los dos primeros elementos: *(hipónimo, hiperónimo)*

Para la construcción se utilizó Natural Language Toolkit (NLTK) ya que consideramos que es la herramienta más práctica y completa para el manejo de OMW y procesamiento de datos léxicos.

3.1. Extracción de pares de hiperonimia

El problema fundamental de la construcción del dataset positivo es que hay escasos recursos para obtener los pares de hiperonimia en español, y no hay un dataset estándar para comparar la calidad del mismo. La extracción se baso en diferentes fuentes y metodologías tradicionales las cuales se detallan a continuación. Posteriormente se evalúan las calidades de los dataset positivos y se emplean diferentes medidas y heurísticas para refinar los mismos.

1. WordNet - OMW:
   Como se mencionó anteriormente WordNet es una base de datos léxica basado en el idioma ingles. El synset es la unidad básica de WordNet que representa un concepto de forma léxica. Cada synset está vinculado a un conjunto de lemmas. Y un lemma es la unidad que representa un sentido específico de cada synset. A su vez por medio de OMW es posible obtener estas lemmas traducidos en español. Entonces considerando cada par de synsets que cumple la relación de hiperonimia, es posible obtener sus lemmas correspondientes en español y construir las tuplas positivos con palabras en español relacionando todas las lemmas de los dos synsets.

2. Basado en patrones:
   Considerando un conjunto de patrones léxicos-sintácticos en español es posible extraer las tuplas positivos a partir de un corpus. Basándose en el trabajo de Ortega [OAVMS11], se consideraron aquellos patrones léxicos-sintácticos en español con sus valores de confianza que se aproximan a 1 como: "el <hipónimo> es el único <hiperónimo>" y "de <hipónimo> y otras <hiperónimo>". Si bien estos patrones dieron resultados positivos sobre un conjunto restringido de catálogos, era interesante para nuestra labor si los mismos patrones muestran mismo desempeño en un corpus grande en español y observar si logran extraer un conjunto de tuplas positivo de calidad superior. Para esto, mediante el uso de las expresiones regulares de los anteriores, se extraen pares de hipónimos e hiperónimos del corpus de Cardellino [Car19]. Posteriormente las tuplas extraídas se filtran usando diferentes medidas como el etiquetado gramatical para mejorar su calidad.

3. Extraídos y traducidos del Dataset de Shwartz:
   En el trabajo de Shwartz [SGD16], los pares de hiperonimia se extraen usando un método híbrido, mediante patrones y múltiples fuentes como WordNet y Wikidata. Su dataset contiene una cantidad considerable de instancias de un cierto conjuntos reducidos de palabras. Por ejemplo las instancias de ciudad, pueblo, compañia como: (sydney, city), (microsoft, company). Considerando solamente aquellos pares de instancias donde los hipónimos son nombres propios que no tienen su traducción correspondiente en español, es posible generar tuplas traduciendo solo la parte de la palabra hiperónimo. Por ejemplo dado la tupla *(Shakespeare, writer)*, traducir solo el hiperónimo y formar la tupla como *(Shakespeare, escritor)*. Para esto fue necesario seleccionar tuplas de instancias con los hipónimos, los nombres propios, que no tengan una traducción correspondientes en español. Entonces limitamos nuestra selección a instancias de: "pueblo", "ciudad", "compañía", "lugar", "río" y "persona". Para la traducción se utilizó la librería googletrans de Google[8].

4. Enlaces transitivos:
   La relación hiperonimia cumple la propiedad transitiva, por lo tanto,

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

es posible considerar los enlaces transitivos como las tuplas positivos. Por ejemplo, dadas tuplas *(perro, animal)* y *(animal, ser_vivo)* es posible extraer el par transitivo *(perro, ser_vivo)*. Para obtener las tuplas transitivas, en primer lugar se genera un grafo dirigido acíclico con los dataset positivos ya extraídos. Al grafo resultante se aplica reducción transitiva para eliminar todos los lazos transitivos redundantes. Luego se obtiene la clausura transitiva del grafo. Finalmente sobre el resultado de la clausura se resta los lazos del grafo original.

3.2. Extracción de pares que no son hiperónimos

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

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.

2. Aleatoria:
   Una forma simple de obtener un par negativo es seleccionar dos palabras de forma aleatoria a partir de sustantivos de vocabulario en español. En primera instancia se construyó un conjunto de sustantivos con los mismos vocabularios de dataset positivo obtenidos de OMW. Posteriormente con el fin de incrementar diversidad de los vocabularios, se expandió el conjunto agregando los sustantivos obtenidos mediante el corpus de Cardellino. Para esto se utilizó la librería Spacy de Python para seleccionar solo las palabras cuyo etiquetado gramatical es sustantivo, y 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 de español.

3. Cohipónimos:
   Como ya se mencionó en la Sección 2.1, la relación de cohiponimia se cumple entre dos hipónimos que tienen el mismo hiperónimo. Y entre dos hipónimos presentan una relación de exclusividad semántica. 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.

4. Antonimia:
   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 y no hay numeroso par de antonimia, es posible extraer un conjunto reducido de tuplas de antonimia en español mediante OMW e incluirlo en el dataset negativo.

3.3. Refinamiento del corpus

Luego de construir la primera versión del dataset, evaluamos su calidad. Para evaluar se extrajo una muestra de cada conjunto de los dataset positivos y negativos, y se verificó manualmente cada tupla si se presentaba o no la relación de hiperonimia entre los pares. La precisión final del dataset que buscábamos era entre $70 \sim 80\,\%$. Sobre aquellos dataset extraídos que presentaban una calidad muy inferior a la precisión final, se aplicó alguna medida heurística y filtración para mejorar su calidad. A continuación se detallan los problemas que deterioraban la calidad de los dataset y las soluciones implementadas para ellos.

3.3.1. Pares extraídos de WordNet - OMW

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 la ambigüedad que se generaba al relacionar las palabras de hiperonimia de origen inglés y obtener los pares traducidos en español mediante OMW. Seguidamente se presentan los problemas del dataset y las soluciones implementadas.

- Orden de la lista de lemmas en español

Dado que cada synset tiene un conjunto de lemmas que varían desde lo más general a lo más específico semánticamente, al considerar todos los lemmas entre los dos synsets de relación de hiperonimia se incrementa la ambigüedad en las tuplas. Por lo tanto, era necesario ordenar la lista de lemmas según su generalidad semántica y luego recortar la lista tomando solo los primeros elementos con significados más genéricos. Wordnet organiza las palabras según una jerarquía donde las palabras más generales ocupan en la parte superior de la jerarquía y las palabras más específicas se encuentran en la parte inferior. Y de la misma los synsets de una palabra y las lemmas de un synset están ordenados. Sin embargo esto no ocurre con OMW. Las lemmas en español de OMW a diferencia de las lemmas de en inglés de WordNet se encuentra ordenado alfabéticamente. Por ejemplo las lemmas de synset "person.n.01" sus lemmas correspondientes en inglés y en español se encuentra ordenado como se muestra en el Cuadro 8.

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

Cuadro 8: Lemmas de "person.n.01"

Entonces una alternativa para ordenar éstos lemmas es ordenar según su frecuencia en un determinado corpus, es decir la cantidad de veces que aparece tal palabra en un corpus. La frecuencia de las palabras de largo mayor a dos se calcula dividiendo la suma de las frecuencias de cada palabra individual por la cantidad de palabras. Para esto, en primera instancia se extrajo la frecuencia desde el corpus español *cess_esp*[9] de 188,650 palabras. Luego para abarcar mayor variedad de vocabularios se extrajo la frecuencia desde el corpus de Cardellino que cuenta

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

con alrededor de 1,5 billones de palabras en español. En el Cuadro 9 se ilustra como quedaron ordenados los lemmas 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 9: Lemmas de"person.n.01" en español ordenadas

Una vez obtenido la lista de lemmas ordenada según su frecuencia generamos diferentes conjuntos de dataset positivo considerando los primeros elementos de la lista para encontrar un conjunto de dataset de calidad aceptable con mayor número de pares posibles. Pues al considerar mayor cantidad de los primeros elementos si bien incrementa el número total de dataset también aumenta la ambigüedad en el mismo.

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

- Filtración según la frecuencia de palabras

Otra medida heurística fue en base la hipótesis planteada en el trabajo de Santus, donde se propone que los hiperónimos son más frecuentes que los hipónimos [SLLS14]. Entonces dado las frecuencias de las palabras obtenidos del corpus de Cardellino, es posible filtrar los pares positivos considerando solo aquellos que cumplen que la frecuencia del hiperónimo sea mayor que la frecuencia del hipónimo.

3.3.2. Pares positivos general

Dado que se cumple la propiedad transitiva en la relación de hiperonimia y además como se presenta ambigüedad en las tuplas, es posible que se presente ciclo en los dataset positivos. Por tanto, se procedió detectar los ciclos sobre todo los dataset extraído y eliminarlos. También detectamos ciertas palabras frecuentes como 'género_de_', 'familia_', etc, los cuales no aportaban ningún valor semántico para el aprendizaje posterior del modelo. Y se procedió normalizar todos los pares tanto los positivos y los negativos.

- El problema de loops y su eliminación

  Para tratar estos casos mediante Networkx, se obtienen las aristas involucradas en ciclos, se quita la primera y se procede a verificar si todavía existen ciclos en los dataset, continuando dicho proceso hasta que no queden ciclos.

  Cabe observar los casos de loops 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 10. Estos casos se eliminaron para reducir la ambigüedad en el dataset.

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

Cuadro 10: Casos de Loops de largo igual a dos

- Normalización de dataset

En algunas palabras con largo mayor a dos aparecen palabras frecuentes que no aportan información semántica como: 'subgénero_de_', 'género_de_', 'subgénero_', 'género_', 'subfamilia_', 'familia_', 'orden_', 'suborden_' y 'ser_'. Estas clases de palabras fueron eliminados y todas las palabras se estandarizaron en minúsculas.

3.4. Evaluación

Para verificar la calidad del dataset, realizamos un muestreo extrayendo de forma aleatoria un conjunto de tuplas de los dataset positivo y negativo. 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*[10]

3.4.1. Muestreo de dataset

A continuación se muestra algunas tuplas extraídas del dataset positivo 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
bimbo	joven	correcto
átomo	fragmento	Incorrecto
servidor	ordenador	Incorrecto
paseo	viaje	correcto
dispositivo	utilaje	correcto
austeridad	rigurosidad	correcto
...	...	...

Cuadro 11: Tuplas extraídas de dataset positivo

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

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 12: Tuplas extraídas de dataset negativo

En el caso de dataset positivo, la aparición de estos errores en su mayoría es debido a la ambigüedad introducida en las lemmas de español de OMW, como átomo-fragmento. Pero consideramos que son casos ambiguos que si bien no se cumple la relación de hiperonimia de forma estricta, sí se presenta un vínculo semántico entre ellos dependiendo del contexto, lo cual no incidirá tanto al aprendizaje del modelo posterior.

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[11]; más la palabra traducida *'edificación'* no contiene el mismo concepto en español.

3.4.2. Resultado de evaluación

Los pares extraídos de WordNet se representa como WN-NaN, donde se considera N primeros elementos de la lista de lemmas luego de aplicar la primera heurística como ya se explicó en la sección 3.3. Como se puede

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

observar en el Cuadro 13, al aplicar la segunda heurística se disminuye la cantidad de 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 13: 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 Caudro 14. El dataset extraído de los lazos transitivos dio un resultado muy inferior a lo esperado y consideramos que es debido al problema de ambigüedad. 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 de los dataset, se definió como el dataset base conformados en parte del positivo WN-3a3 aplicado ambas medidas 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	$< 100\ 000$

Negativos	Invertidos	Aleatoria	Cohipónimos	Antonimos
Precisión	85 %	100 %	100 %	90 %
Cantidad	$\sim$ # WN-NaN	~ 90000	~ 45000	1107

Cuadro 14: Resultado del muestreo

3.5. Partición del corpus

En esta sección se detalla como se dividió todo el conjunto de dataset (tuplas positivo y negativo) para el entrenamiento del modelo. Como otro método supervisado usual, la partición se realiza en tres conjuntos: entrenamiento, validación y test. Más siguiendo el trabajo de Shwartz se consideró dos particiones diferentes: Partición aleatoria y partición lexical [SGD16]. En la siguiente sección se detalla cada uno.

3.5.1. Partición aleatoria

El método también conocido como *random split* consiste en dividir los dataset de forma aleatoria con una distribución uniforme entre los 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 15.

3.5.2. Partición sin intersección léxica

El problema que puede presentar en el proceso de entrenamiento del modelo es 'lexical memorization', ya mencionado en la sección 2.6.2. Para evitar este fenómeno, se realiza *lexical split* donde se divide los conjuntos con intersección de términos nula. En otras palabras, cada conjunto contiene un vocabulario distinto, para evitar que el modelo se sobre-ajuste con la memorización léxica [LRBD15]. Además, el entrenamiento de un modelo en un dataset dividido sin intersección léxica, puede dar como resultado un modelo más general, que puede manejar mejor las tuplas con dos vocabularios nuevos durante el proceso de inferencia. Para implementación de este algoritmo se baso en el trabajo de Shwartz [SGD16], donde se enfoca dividir los tres conjuntos, entrenamiento, validación y test con una proporción aproximada de 70/5/25 %. La proporción de lexical split se muestra en el Cuadro 15.

		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 15: Tamaño de las particiones

4. Modelo de detección de hiperonimia

El objetivo de esta sección es construir modelos de redes neuronales y entrenarlos con diferentes combinaciones de dataset con el fin de obtener un modelo riguroso 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 las entradas para los modelos. Ya teniendo las particiones de los dataset (Sección 3.5) y sus vectores correspondientes se entrenan los modelos sobre el conjunto de entrenamiento. Y con el conjunto de validación se selecciona el mejor de los modelos entrenados y se ajustan los parámetros buscando la configuración óptima. Finalmente se evalúa el mejor de los modelos sobre el conjunto de prueba, lo que nos permite detectar el error real cometido y la precisión del modelo para la detección de hiperonimia en español. En la figura 9 se muestra un bosquejo simplificado de este procedimiento.

En las siguientes subsecciones detallaremos sobre el método de Word Embeddings que empleamos para obtener los vectores de las palabras, los modelos neuronales construidos, los procesos de entrenamiento y evaluación, y finalmente los resultados que obtuvimos con cada uno de los modelos seleccionados.

4.1. Word embeddings

A pesar de que numerosos métodos de Word Embeddings han sido desarrollados, para la representación vectorial de palabras en idioma español hay escasos recursos. Dentro de aquellos que son considerados con alta calidad de cobertura se encuentra el corpus de Cardellino [Car19] 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 2.6.2). También

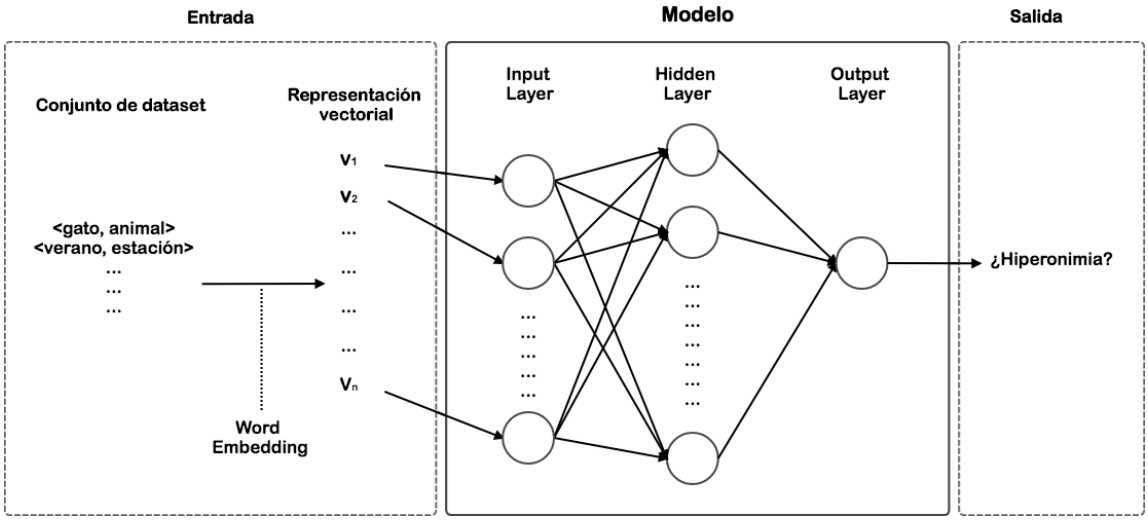


Figura 9: Bosquejo simplificado

fastText de Common Crawl[12] ofrece vectores de palabras preentrenados para distintos lenguajes, donde los vectores son entrenados sobre los corpus de Wikipedia [BGJM17]; [GBGJM18]. Nosotros hemos optado por utilizar fasttext de Common Crawl ya que la misma se caracteriza por su cobertura amplia sobre cualquier vocabulario de idioma.

Una vez que se obtienen los dos vectores de un par de palabras del dataset, es necesario combinar estos dos para pasarlo como entrada del modelo de red neuronal. Ya existen diversos trabajos donde realizan distintas operaciones aritméticas sobre estos vectores embeddings por ejemplo: la concatenación de vectores *(*$x \oplus y$*)* [BBDS12], la diferencia y-x [REB14], y el producto escalar $x \cdot y$, los cuales junto con el modelo de red neuronal muestran buenos resultados [WW03].

4.2. Modelo red neuronal

Como se mencionó anteriormente, en base a un enfoque supervisado, se entrenan los modelos de redes neuronales con los vectores embeddings de palabras. El primer modelo se construyó en base los vectores combinados

[12]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/`

mediante la concatenación de los mismos *(x ⊕ y)* [BBDS12]. Posteriormente se agregó un nuevo modelo basado en Order Embedding con un enfoque diferente para entrenar el modelo [VKFU15]. A continuación detallaremos sobre las características de dichos modelos como su arquitectura, su función de pérdida y otros detalles.

- Modelo con la concatenación de vectores:

  Por un lado se siguió el propuesto de Baroni [BBDS12], donde se concatenan los dos vectores 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 de 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 secuencial, donde las capas son ordenados de forma lineal. Como el problema es de clasificación binaria para la función de pédida se elegió *binary cross entropy*.

  La configuración de los demás parámetros como la cantidad capas, la cantidad de neuronas por capa, función de activación, función de optimización, función de pérdida, etc se detallan más adelante cuando se estudia los parámetros óptimos del modelo.

- El Modelo Order Embedding:

  Es el modelo propuesto por Vendrov [VKFU15], donde a diferencia de los métodos tradicionales, en lugar de combinar los dos vectores de pares de palabras para entrenar sobre un modelo, emplea un método para entrenar Order Embeddings en $\Re^m_{\geq 0}$ considerando *reversed product order* como se muestra a continuación:

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

Donde $x, y \in \Re^m_{\geq 0}$, x_i y y_i corresponden al componente i-ésimo de x e y, respectivamente. Por definición, esta relación es antisimétrica y transitiva y $\vec{0}$ es el elemento superior de la jerarquía.

La relación de orden parcial $(\preceq, \Re^m_{\geq 0})$ definida anteriormente permite definir medidas para cuantificar el grado en que un par de dos elementos no satisface la relación. Dejenos considerar:

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

donde $\vec{x}, \vec{y} \in \Re^m_+$ y max es el máximo de sus componentes. Notar que E_p indica el grado de no satisfacción de la relación y $E_p(x, y) = 0$ si $\vec{x} \preceq \vec{y}$.

Además, E_p es forzada a ser mayor que un umbral α para términos no relacionados a través de la función *max-margin loss*, como sigue:

$$E_n(\vec{x}, \vec{y}) = max\{0, \alpha - E_p(\vec{x}, \vec{y})\}. \tag{3}$$

Notar que $E_n(\vec{x'}, \vec{y'})$ es 0 cuando $E_p(\vec{x'}, \vec{y'}) \geq \alpha$ garantiza que $\vec{x'}\vec{y'}$. Luego, sumando (2) y (3), la función de pérdida de contraste resultante, que consiste en minimizar E_p y E_n conjuntamente, es la siguiente:

$$L = \sum_{(x,y)\in P} E_p(\vec{x}, \vec{y}) + \sum_{(x',y')\in N} E_n(\vec{x'}, \vec{y'}), \tag{4}$$

donde P y N son conjuntos de ejemplos positivos y negativos, respectivamente. Tener en cuenta que L es diferenciable, lo que permite entrenar un order embedding a través del descenso por gradiente (y variantes estocásticas o mini lotes).

Notar que de acuerdo con esta función de pérdida, si se usa una red neuronal para el mapeo, entonces el modelo puede interpretarse como una red siamesa con una medida de distancia asimétrica (Ver la Figura 10).

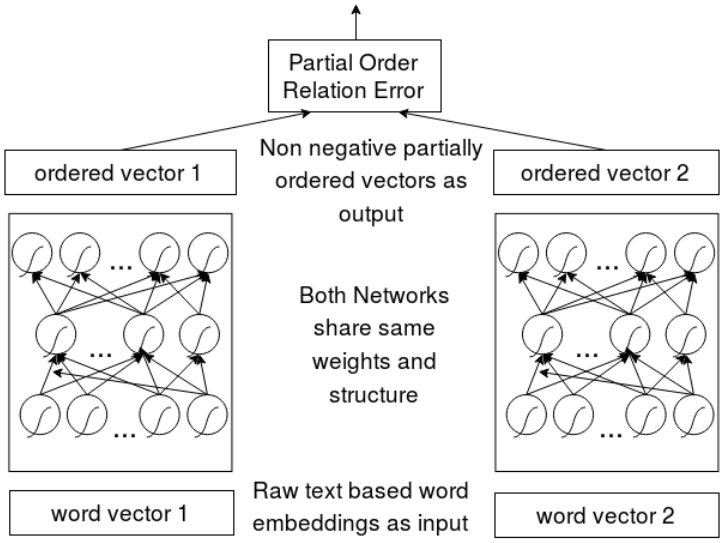


Figura 10: Order embedding.

4.3. Implementación y entrenamiento

Para el manejo de RNAs se utilizó la librería Keras [Cho], la cual proporciona un API de redes neuronales de alto nivel, escrita en Python y capaz de ejecutarse sobre TensorFlow [Goo], CNTK [Mic] o Theano [MIL]. Keras permite realizar prototipos de forma fácil y rápida, a través de la facilidad de uso, modularidad y extensibilidad. Utilizamos TensorFlow como el motor backend de Keras, el cual es un framework open-source desarrollado por Google.

Mediante la librería de Keras se definieron parámetros estructurales del modelo como cantidad y tamaño de capas, y las funciones de activación de las neuronas. Con lo cual realizamos pruebas con distintas cantidad de capas y neuronas por capa, además se utilizaron diferentes funciones tanto de activación como de optimización para poder lograr la mejor combinación posible de parámetros para nuestro modelo.

Para evaluar el desempeño de los modelos entrenados se utilizaron las tres métricas básicas de clasificación binaria: *precision*, *recall* y *medida F1*. (Ver la Sección 2.5.3)

4.3.1. Búsqueda de hiperparámetros

Para encontrar configuración óptima de los modelos, se utilizaron métodos **Grid Search** [scia] y **Random Search** [scib]. Estas dos técnicas son principales para encontrar un mejor ajuste de los **hiperparámetros**. Los hiperparámetros son los mismos parámetros que configuran el modelo de red neuronal que se ajustan con el fin de encontrar una configuración óptima para el aprendizaje del modelo. Si bien no se utilizan para modelar los datos directamente, pero influyen en la capacidad y características de aprendizaje del modelo.

En primer lugar se definieron los hiperparámetros que se consideraron más importantes para los dos modelos:

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

Grid search es una forma tradicional de realizar la optimización del hiperparámetros, y funciona buscando exhaustivamente a través de un subconjunto específico de hiperparámetros. En otras palabras, se evalúa el modelo para todas las combinaciones posibles sobre el subconjunto definido. Si bien es una técnica que permite encontrar la configuración óptima de los hiperparámetros, el tiempo de procesamiento es exponencial al tamaño del subconjunto.

Random Search difiere de Grid Search principalmente en que busca el subconjunto especificado de hiperparámetros aleatoriamente en lugar de exhaustivamente. El mayor beneficio es la disminución del tiempo de procesamiento. Si bien no garantiza una configuración óptima de hiperparámetros, permite distinguir aquellos hiperparámetros que no afectan significativamente al resultado de los modelos y aquellos que favorecen directamente al resultado.

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.

4.3.2. 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[13], se intentó encontrar los hiperparámetros para el modelo. Sin embargo, nos encontramos con problemas debido a grandes demoras de tiempos de ejecución (4 a 5 días sin lograr finalizar), ya que para poder explorar varios de los hiperparámetros que consideramos importantes, no se logró que termine de ejecutar el Grid Search. Sucedía que luego de ejecutarse por varios días se interrumpía la misma sin poder saber la razón específica por la cual se interrumpe. Por tal motivo optamos por utilizar Random Search en lugar de Grid Search.

Para Random Search en primer lugar se intentó utilizar 1000 combinaciones pero el tiempo de computación era muy grande y demoraba días sin finalizar. Con lo que decidimos bajar a 150 combinaciones y al cabo de un poco más de un día finalizó la ejecución. A continuación de detalla la configuración de los hiperparámetros encontrados según Random Search:

- Cantidad de capas ocultas = 2
- Primer 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*

[13]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/`

- Optimizador: *adam*
- En el entrenamiento se realizaron *20 epochs* con *batch_size de 32*

Adicionalmente, se utilizó el callback **EarlyStopping** [Ker], el cual detiene el entrenamiento cuando una medida monitoreada deja de mejorar por cierto tiempo o intervalos. La medida monitoreada fue la ya mencionada *F1*, y se pretende una mejora mínima de un 1 % cada 5 *epochs* (*min_delta = 0.01 y patience = 5*), de lo contrario el entrenamiento se detiene.

4.3.3. Modelo Order Embedding

Se aplicó la técnica de Random Search sobre algunos de los parámetros que se consideró más relevantes. Y la configuración resultante fue la siguiente:

- Cantidad de capas ocultas = 3
- Primer 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*

5. Resultados

En esta sección se muestra los resultados obtenidos de entrenar los modelos descritos anteriormente con los dataset en español. Para evaluar los modelos se consideraron las tres métricas ya mencionadas: *precision*, *recall* y *medida F1*. El entrenamiento del modelo se realizaron con diferentes combinaciones del dataset con el fin de observar como varía el resultado del aprendizaje del modelo con diferentes subconjuntos de dataset positivo y negativo. A continuación se muestran las combinaciones de los dataset:

- Dataset base definido en la Sección 3.4.2, esta conformado en parte de dataset positivo, tuplas WN-3a3 aplicadas las medidas heurísticas y las traducidas de Shwartz, y en parte de dataset negativo, las tuplas extraídas de forma aleatoria a partir de los vocabularios de WordNet y Cardellino y por último los antónimos. (Base)
- Al dataset base se le agregan el conjunto de cohipónimos como instancias negativas para el entrenamiento. (Base +cohyp)
- Al dataset base se le agregan las instancias positivas extraídas por patrones léxicas del corpus de Cardellino. (Base +pattern)
- Al dataset base, además de agregar el dataset pattern, se le agregan el conjunto de cohipónimos como instancias negativas para el entrenamiento. (Base +pattern +cohyp)

Además se evaluaron los resultados en paralela sobre la partición de test agregando las instancias de los cohipónimos. Es decir, evaluar el modelo entrenado 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.

5.1. Resultados de modelo con concatenación de vectores

Los resultados obtenidos del modelo de concatenación se muestra en el Cuadro 16. Como se puede apreciar en el cuadro, tanto a la partición aleatoria y lexical, al incluir las instancias obtenidas de los patrones léxicas y las instancias de cohipónimos mejora el resultado de la medida F_1. Más al evaluar

el modelo sobre la partición de test que incluye las instancias de cohipónimos no se observó una mejora considerable.

	P_{rand}	R_{rand}	F_{rand}	P_{lex}	R_{lex}	F_{lex}
Base	0.881	0.820	0.839	0.756	0.775	0.753
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**

	P_{rand}	R_{rand}	F_{rand}	P_{lex}	R_{lex}	F_{lex}
Base	0.806	0.850	0.818	0.760	**0.875**	**0.803**
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 16: Resultados del Modelo de concatenación. La tabla superior es el resultado de considerar las instancias de cohipónimos solamente en la partición de entrenamiento y la tabla inferior es al considerar las instancias cohipónimos en la partición de entrenamiento y prueba.

5.2. Resultados de modelo Order Embedding

Los resultados obtenidos del modelo de Order Embedding se muestran en el Cuadro 17. Como se puede observar en los resultados, tanto las instancias de cohipónimos como las extraídas de patrones durante el entrenamiento proporcionan alguna mejora. A diferencia del modelo de concatenación, en su mayoría al considerar solo las instancias de cohipónimos sobre la base ha dado mejor resultado, con la excepción del caso de lexical split con los cohipónimos agregados sobre la partición de prueba.

	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

	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 17: Resultados del Modelo Order Embedding. La tabla superior es el resultado de considerar las instancias de cohipónimos solamente en la partición de entrenamiento y la tabla inferior es al considerar las instancias cohipónimos en la partición de entrenamiento y prueba.

6. Conclusión y trabajo futuro

En éste 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 sus etapas de Entrenamiento, Validación y Testing, y la intersección léxica entre ellos: 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 de 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 cohimponimos y pares extraídos con la utilización de patrones durante el entrenamiento sin mejoras considerables.

Este trabajo de investigación consiguió resultados interesantes. No obs-

tante, existen varios aspectos sobre los cuales se desea trabajar a futuro, entre ellos se tienen:

- Enriquecer nuestro dataset agregando nuevos vocabularios de algunas áreas específicas, ya sea relacionado al terminologia medica o industrial, etc, con métodos tradicionales para la detección de hiperonimia.
- Abordar el problema de ambigüedad con un mecanismo formal ya sea para filtrar los casos falsos positivos o mejorar la generación de pares de hiperonimia a partir de OMW.
- Realizar comparación sistemático de los dataset obtenidos con WordNets existentes de Español y establecer una medida para definir la calidad de dataset de hiperonimia en Español.
- Estudiar la existencia del aprendizaje del modelo entre dos lenguajes distintos entrenando el modelo con un dataset bilinbgüe.
- Extender el trabajo para la detección de otra relación léxica.

Referencias

[Har54] Zellig S. Harris. "Distributional Structure". En: (1954), págs. 146-162. DOI: 10.1080/00437956.1954.11659520.

[Hin86] Geoffrey E. Hinton. "Learning distributed representations of concepts". En: (dic. de 1986).

[DFLDH88] S. T. Dumais y col. "Using Latent Semantic Analysis to Improve Access to Textual Information". En: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI '88. Washington, D.C., USA: ACM, 1988, págs. 281-285. ISBN: 0-201-14237-6. DOI: 10.1145/57167.57214. URL: http://doi.acm.org/10.1145/57167.57214.

[MBFGM91] George Miller y col. "Introduction to WordNet: An On-line Lexical Database*". En: 3 (ene. de 1991). DOI: 10.1093/ijl/3.4.235.

[Hea92] Marti A. Hearst. "Automatic Acquisition of Hyponyms from Large Text Corpora". En: *14th International Conference on Computational Linguistics, COLING 1992, Nantes, France, August 23-28, 1992*. 1992, págs. 539-545. URL: http://aclweb.org/anthology/C92-2082.

[LBA95] Kevin Lund, Curt Burgess y Ruth Atchley. "Semantic and associative priming in high-dimensional semantic space". En: ene. de 1995, págs. 660-665.

[Vos98] Piek Vossen. "Introduction to EuroWordNet". En: *EuroWordNet: A multilingual database with lexical semantic networks*. Ed. por Piek Vossen. Dordrecht: Springer Netherlands, 1998, págs. 1-17. 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.

[Roc00] Salvador Climent Roca. "Individuación e información Parte-Todo; Representación para el procesamiento computacional del lenguaje)". En: 8 (2000).

[PBG02] Emanuele Pianta, Luisa Bentivogli y C. Girardi. "MultiWordNet: Developing an Aligned Multilingual Database". En: (ene. de 2002).

[BDVJ03] Yoshua Bengio y col. “A Neural Probabilistic Language Model”. En: *J. Mach. Learn. Res.* 3 (mar. de 2003), págs. 1137-1155. ISSN: 1532-4435. URL: http://dl.acm.org/citation.cfm?id=944919.944966.

[DPO03] J. Daudé, Lluís Padró y Gemma Oliver. “Making Wordnet Mappings Robust”. En: *Procesamiento del lenguaje natural, ISSN 1135-5948, Nº. 31, 2003, pags. 47-54* (ene. de 2003).

[Rap03] Reinhard Rapp. “Word sense discovery based on sense descriptor dissimilarity”. En: (dic. de 2003).

[WW03] Julie Weeds y David Weir. “A General Framework for Distributional Similarity”. En: *Proceedings of the 2003 Conference on Empirical Methods in Natural Language Processing*. 2003, págs. 81-88. URL: https://www.aclweb.org/anthology/W03-1011.

[Wee04] Weir D. McCarthy D. Weeds J. “Characterising measures of lexical distri- butional similarity”. En: (2004).

[PM05] Philipp Cimiano Paul Buitelaar y Bernado Magnini. “Ontology Learning from Text: Methods, Evaluation and Applications”. En: (jun. de 2005), pág. 180.

[SJN05] Rion Snow, Daniel Jurafsky y Andrew Y. Ng. “Learning Syntactic Patterns for Automatic Hypernym Discovery”. En: *Advances in Neural Information Processing Systems 17*. Ed. por L. K. Saul, Y. Weiss y L. Bottou. MIT Press, 2005, págs. 1297-1304. URL: http://papers.nips.cc/paper/2659-learning-syntactic-patterns-for-automatic-hypernym-discovery.pdf.

[DMVH06] Claudia Denicia-Carral y col. “A Text Mining Approach for Definition Question Answering”. En: *Advances in Natural Language Processing*. Ed. por Tapio Salakoski y col. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, págs. 76-86. ISBN: 978-3-540-37336-0.

[SJN06] Rion Snow, Daniel Jurafsky y Andrew Ng. “Semantic Taxonomy Induction from Heterogenous Evidence.” En: ene. de 2006. DOI: 10.3115/1220175.1220276.

[RA07] Elena Rambla y Amparo Alcina Caudet. “La relación hiponimia-hiperonimia en los términos de la cerámica industrial”. En: 3 (ene. de 2007). DOI: 10.1093/ijl/3.4.235.

[SAAB08] Gerardo Sierra y col. “Definitional verbal patterns for semantic relation extraction”. Inglés. En: *Terminology* 14.1 (ago. de 2008), págs. 74-98. ISSN: 0929-9971. DOI: 10.1075/term.14.1.05sie.

[FL10] Yansong Feng y Mirella Lapata. “Visual Information in Semantic Representation”. En: *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*. Los Angeles, California: Association for Computational Linguistics, jun. de 2010, págs. 91-99. URL: https://www.aclweb.org/anthology/N10-1011.

[Mon10] Ana María Fernández Montraveta. “La construcción del WordNet 3.0 en español”. En: (ene. de 2010), págs. 201-220.

[TP10] P. D. Turney y P. Pantel. “From Frequency to Meaning: Vector Space Models of Semantics”. En: *Journal of Artificial Intelligence Research* 37 (feb. de 2010), págs. 141-188. ISSN: 1076-9757. DOI: 10.1613/jair.2934. URL: http://dx.doi.org/10.1613/jair.2934.

[OAVMS11] Rosa María Ortega y col. “Hacia la identificación de relaciones de hiponimia/hiperonimia en Internet”. es. En: *Revista signos* 44 (mar. de 2011), págs. 68-84. ISSN: 0718-0934. URL: https://scielo.conicyt.cl/scielo.php?script=sci_arttext&pid=S0718-09342011000100006&nrm=iso.

[BBDS12] Marco Baroni y col. “Entailment Above the Word Level in Distributional Semantics”. En: *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*. EACL ’12. Avignon, France: Association for Computational Linguistics, 2012, págs. 23-32. ISBN: 978-1-937284-19-0. URL: http://dl.acm.org/citation.cfm?id=2380816.2380822.

[GLR12] Aitor Gonzalez-Agirre, Egoitz Laparra y German Rigau. "Multilingual Central Repository version 3.0". En: *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC'12)*. Istanbul, Turkey: European Language Resources Association (ELRA), mayo de 2012, págs. 2525-2529. URL: http://www.lrec-conf.org/proceedings/lrec2012/pdf/293_Paper.pdf.

[BF13] Francis Bond y Ryan Foster. "Linking and Extending an Open Multilingual Wordnet". En: *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Sofia, Bulgaria: Association for Computational Linguistics, ago. de 2013, págs. 1352-1362. URL: https://www.aclweb.org/anthology/P13-1133.

[MCCD13] Tomas Mikolov y col. *Efficient Estimation of Word Representations in Vector Space*. 2013. arXiv: 1301.3781 [cs.CL].

[BDK14] Marco Baroni, Georgiana Dinu y Germán Kruszewski. "Don't count, predict! A systematic comparison of context-counting vs. context-predicting semantic vectors". En: *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Baltimore, Maryland: Association for Computational Linguistics, jun. de 2014, págs. 238-247. DOI: 10.3115/v1/P14-1023. URL: https://www.aclweb.org/anthology/P14-1023.

[PSM14] Jeffrey Pennington, Richard Socher y Christopher Manning. "Glove: Global Vectors for Word Representation". En: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Doha, Qatar: Association for Computational Linguistics, oct. de 2014, págs. 1532-1543. DOI: 10.3115/v1/D14-1162. URL: https://www.aclweb.org/anthology/D14-1162.

[REB14] Stephen Roller, Katrin Erk y Gemma Boleda. "Inclusive yet Selective: Supervised Distributional Hypernymy Detection". En: *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*. Dublin, Ireland: Dublin City University y Association for Compu-

tational Linguistics, ago. de 2014, págs. 1025-1036. URL: https://www.aclweb.org/anthology/C14-1097.

[SLLS14] Enrico Santus y col. "Chasing Hypernyms in Vector Spaces with Entropy". En: *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics, volume 2: Short Papers*. Gothenburg, Sweden: Association for Computational Linguistics, abr. de 2014, págs. 38-42. DOI: 10.3115/v1/E14-4008. URL: https://www.aclweb.org/anthology/E14-4008.

[LRBD15] Omer Levy y col. "Do Supervised Distributional Methods Really Learn Lexical Inference Relations?" En: *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Denver, Colorado: Association for Computational Linguistics, mayo de 2015, págs. 970-976. DOI: 10.3115/v1/N15-1098. URL: https://www.aclweb.org/anthology/N15-1098.

[LM15] Alessandro Lopopolo y Emiel van Miltenburg. "Sound-based distributional models". En: *Proceedings of the 11th International Conference on Computational Semantics*. London, UK: Association for Computational Linguistics, abr. de 2015, págs. 70-75. URL: https://www.aclweb.org/anthology/W15-0110.

[VKFU15] Ivan Vendrov y col. "Order-Embeddings of Images and Language". En: *CoRR* abs/1511.06361 (2015). arXiv: 1511.06361. URL: http://arxiv.org/abs/1511.06361.

[SGD16] Vered Shwartz, Yoav Goldberg e Ido Dagan. "Improving Hypernymy Detection with an Integrated Path-based and Distributional Method". En: *CoRR* abs/1603.06076 (2016). arXiv: 1603.06076. URL: http://arxiv.org/abs/1603.06076.

[BGJM17] Piotr Bojanowski y col. "Enriching Word Vectors with Subword Information". En: *Transactions of the Association for Computational Linguistics* 5 (2017), págs. 135-146. DOI: 10.1162/tacl_a_00051. URL: https://www.aclweb.org/anthology/Q17-1010.

[MGBPJ17] Tomas Mikolov y col. *Advances in Pre-Training Distributed Word Representations*. 2017. arXiv: 1712.09405 [cs.CL].

[Gar18] Iker García Ferrero. "Estudio de word embeddings y métodos de generación de meta embeddings". En: (2018).

[GBGJM18] Edouard Grave y col. *Learning Word Vectors for 157 Languages*. 2018. arXiv: 1802.06893 [cs.CL].

[JD18] Sergio Jimenez y 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". En: nov. de 2018, págs. 392-403. ISBN: 978-3-030-03927-1. DOI: 10.1007/978-3-030-03928-8_32.

[Car19] Cristian Cardellino. *Spanish Billion Words Corpus and Embeddings*. Ago. de 2019. URL: https://crscardellino.github.io/SBWCE/.

[Bal] Alfonso Ballesteros. *Desarrollo de un Marco de Trabajo para el diseño de Redes Neuronales en Java: JRedesNeuronales*. URL: http://www.redes-neuronales.com.es/.

[Cen] 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.

[Cho] François Chollet. *Keras: The Python Deep Learning library*. URL: https://keras.io/.

[Coe] Fabián Coelho. *Significado de Sinónimo*. URL: https://www.significados.com/sinonimo/.

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

[Ker] Keras. *Usage of callbacks*. URL: https://keras.io/callbacks/#earlystopping.

[Mic] Microsoft. *The Microsoft Cognitive Toolkit*. URL: https://docs.microsoft.com/es-es/cognitive-toolkit/.

[MIL] MILA. *Theano*. URL: http://www.deeplearning.net/software/theano/.

[scia] scikit-learn.org. *sklearn.model$_s$election.GridSearchCV*. URL: https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html.

[scib] scikit-learn.org. *sklearn.model_selection.RandomizedSearchCV*. URL: https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.RandomizedSearchCV.html.