UNIVERSIDAD DE LA REPÚBLICA

FACULTAD DE INGENIERÍA

Instituto de Computación- InCo

PROYECTO DE GRADO

Inducción del Sentido de las Palabras para el Idioma Español

Autor:
Rodrigo Lastra

Tutores:
Luis Chiruzzo
Mathias Etcheverry

Montevideo, Uruguay
2019

RESUMEN

En cualquier lenguaje natural, existe una gran cantidad de palabras con más de un significado posible. Las máquinas necesitan procesar información textual no estructurada y transformarla en estructuras que puedan analizarse para determinar los significados de las palabras. El problema de identificar, computacionalmente, el significado de una o más palabras en su contexto es conocido como *Word Sense Disambiguation* (WSD).
Para la resolución de WSD es necesario contar con inventarios de significados utilizables computacionalmente de calidad para el lenguaje a desambiguar, que sean a la vez relevantes para el dominio de aplicación en el que se busca realizar la desambiguación. A su vez la construcción de sistemas de aprendizaje automático supervisado o basados en conocimiento requieren otros tipos de recursos lingüísticos (tesauros, ontologías, etc.). Tanto los inventarios de significados como este otro tipo de recursos son costosos de construir en tiempo y recursos, y son dependientes del dominio de aplicación y el lenguaje. Para evitarlo, se recurre a técnicas de aprendizaje no supervisado que descubren automáticamente los posibles significados de una palabra, a partir de corpus sin anotar. Este problema se conoce como *Word Sense Induction* (WSI). Lamentablemente, en español los esfuerzos en WSI han sido prácticamente inexistentes.
Este trabajo, busca aplicar en español una técnica de WSI denominada "*WSI with neural biLM and symmetric patterns*" basada en un modelo de lenguaje neuronal bidireccional biLM ELMo que obtuvo un rendimiento superior a los previamente reportados para la tarea de referencia SemEval 2013 *Task 13* para el idioma inglés.
La evaluación del método de desambiguación en español se realizará sobre la tarea Senseval 2 *Spanish Lexical Sample*.
Para la aplicación del método de desambiguación, fue necesario realizar el entrenamiento del modelo de lenguaje biLM ELMo para el español, del cual pueden obtenerse *embeddings* con o sin contexto, así como ser utilizado como un modelo de lenguaje completo. Se trata de un recurso que no se encontraba disponible previamente en forma completa y que se encuentra disponible públicamente[1].

Palabras clave:
Procesamiento de lenguaje natural (NLP), Word Sense Disambiguation (WSD), knowledge acquisition bottleneck, Word Sense Induction (WSI), no supervisado, ELMo, symmetric patterns, modelo de lenguaje neuronal, embeddings

[1] https://github.com/lastrodrigo/SpanishElmoFullWeights

Contenido

1.Introducción

El lenguaje humano está cargado de ambigüedades. Gran cantidad de palabras pueden ser interpretadas de distintas formas de acuerdo al contexto en el que son utilizadas. Tomemos por ejemplo las siguientes oraciones:

A. Depositó el dinero en el *banco*.
B. El parque tiene un *banco* frente al estanque.

Para cualquier hablante del español resulta evidente que las ocurrencias de la palabra *banco* en las dos oraciones hacen referencia a significados diferentes: por un lado, una empresa dedicada a realizar operaciones financieras, y por otro, un asiento.
La mente humana está acostumbrada a tratar con este tipo de situaciones y por tanto, la mayor parte del tiempo los humanos no somos conscientes de las ambigüedades del lenguaje. Las máquinas en cambio, necesitan procesar información textual no estructurada y transformarla en estructuras que puedan analizarse para determinar los significados subyacentes de las palabras [Navigli, 2012].
El problema de identificar, computacionalmente, el significado de una o más palabras en su contexto es conocido con el nombre en inglés *Word Sense Disambiguation* (abreviado WSD). Se trata de un problema de inteligencia artificial "*AI-complete*" [Ide y Véronis, 1998], es decir, un problema de dificultad equivalente a resolver los problemas centrales de la inteligencia artificial, como por ejemplo el Test de Turing [Yampolskiy, 2013].
En el ejemplo anterior, sería esperable que un algoritmo para resolver WSD cuente, por ejemplo, con el diccionario de la Real Academia Española como inventario de significados, donde la palabra "banco" se define como:

1. Asiento, con respaldo o sin él, en que pueden sentarse dos o más personas.

2. Madero grueso escuadrado que se coloca horizontalmente sobre cuatro pies y sirve de mesa para labores de carpinteros y otros artesanos.

3. En los mares, ríos y lagos navegables, bajo que se prolonga en una gran extensión.

4. Conjunto de peces que van juntos en gran número.

5. Empresa dedicada a realizar operaciones financieras con el dinero procedente de sus accionistas y de los depósitos de sus clientes.

6. Establecimiento médico donde se conservan y almacenan órganos, tejidos o líquidos fisiológicos humanos para cubrir necesidades quirúrgicas, de investigación, etc. Banco de ojos, de sangre.

...

El algoritmo debería entonces asignarle a la ocurrencia de *banco* en la oración A el significado número 5, y a la ocurrencia de *banco* en B el significado número 1.
Sin embargo, los inventarios de significados presentan algunos aspectos desfavorables. Por un lado necesitan ser continuamente actualizados. Además, suelen tener en cuenta solamente significados lexicográficos (por ejemplo, "león" como animal) e ignorar nombres propios (por ejemplo, "León" como ciudad en España). En el caso de algunos idiomas para los que no se cuenta con suficientes recursos lingüísticos, no existen inventarios de significados de calidad utilizables computacionalmente que permitan aplicar este enfoque. Afortunadamente este no es el caso de nuestro idioma, aún cuando las versiones en español de algunos recursos muy utilizados (como WordNet [Miller et al, 1990]) no sean completas. Esta dificultad se presenta también en caso que se desee implementar WSD para un dominio altamente específico (por ejemplo: medicina u otros ámbitos científicos especializados) donde los significados de ciertas palabras pueden diferir de los recogidos del uso general del idioma por los inventarios de significados existentes.
Además de los inventarios de significados, la construcción de sistemas de aprendizaje automático capaces de realizar WSD con enfoques supervisados requiere de corpus (colecciones de texto) anotados de gran extensión. Los sistemas basados en conocimiento, utilizan otros tipos de recursos lingüísticos como tesauros y ontologías. Todas estas fuentes de conocimiento son extremadamente costosas de construir en cuanto a tiempo y recursos requeridos, y deben rehacerse en cada ocasión que se modifica el escenario en el que se pretende aplicar WSD (nuevos dominios, diferentes lenguajes o inventarios de significados). Este es un problema persistente en WSD, llamado *knowledge acquisition bottleneck* ("cuello de botella de adquisición de conocimiento") [Ide y Véronis, 1998].
Para superar las dificultades planteadas por WSD en cuanto al mencionado *knowledge acquisition bottleneck*, se recurre a técnicas que apuntan al descubrimiento automático de los posibles significados de una palabra mediante aprendizaje no supervisado, haciendo uso únicamente de corpus sin anotar, es decir sin ningún tipo de información sobre los significados que puede tener una palabra. A este problema se le denomina *Word Sense Induction* (abreviado como WSI).
Supongamos que, continuando con el ejemplo original, queremos mediante WSI diferenciar los significados de "banco" en las oraciones A y B. Para ello será necesario contar con múltiples oraciones conteniendo la palabra "banco", además de nuestras oraciones originales. Por ejemplo, digamos que se tienen las oraciones:

A. Depositó el dinero en el *banco*.
B. El parque tiene un *banco* frente al estanque.
C. Tiroteo, persecución y un delincuente detenido por el robo a un *banco* en Quilmes.
D. Algunos también se sientan y fotografían el *banco*.
E. En cuanto al *banco* de sangre, refirió que tienen la capacidad de mil unidades de sangre.

El resultado esperado, sería que las oraciones A y C sean agrupadas juntas, y lo mismo para las oraciones B y D, mientras que E no comparte significado con ninguna otra. Podemos inferir, como hablantes del español, que el grupo formado por A y C corresponde al significado 5 de la RAE, mientras que el de B y D corresponde al significado 1, y el

formado por E al significado 6, pero esto no es conocido para el método de WSI, que solamente agrupa instancias que comparten significado, sin conocer de qué significado se trata.
Complejizando un poco más el problema de WSI, podemos tener WSI ponderado, que es el problema que resuelve el método de desambiguación estudiado en este trabajo, donde a cada instancia a desambiguar se le asigna una probabilidad de pertenencia a cada agrupamiento (significado). En el caso del ejemplo, supongamos que el método es suficientemente sensible como para diferenciar el significado de "banco" como empresa financiera inmaterial, del edificio o sucursal física donde esta desarrolla sus actividades. Notoriamente la diferencia es sutil, e incluso en el diccionario de la RAE no existen dos significados distintos para estos casos, pero el método podría llegar a considerar que la oración C se ajusta en un alto grado (digamos 0.80, por ejemplificar) al conjunto donde se agrupan las instancias que referencian al edificio material de la sucursal bancaria, y en un menor grado a aquel que agrupa las referencias a la empresa financiera en forma abstracta (con una ponderación de 0.20, por continuar con el ejemplo[2]), mientras que para el caso de la oración A se podría pensar que se da un menor ajuste al primero de los conjuntos mencionados y mayor al segundo (digamos, 0.25 y 0.75).

1.1. Motivación

Desde los comienzos de este siglo, múltiples trabajos han atacado el problema de WSD para el idioma español, en particular en el marco de las evaluaciones internacionales Senseval y SemEval (sobre las que se profundizará más adelante). Sin embargo, en lo que refiere a WSI en español los esfuerzos han sido prácticamente inexistentes, quizás en parte por la ausencia de evaluaciones internacionales que los promuevan. Es por esto, que la finalidad de este trabajo es presentar resultados para WSI en español utilizando métodos actuales que alcanzaron los mejores resultados reportados en las tareas de referencia para el inglés. En particular, se trabajará en la adaptación del método publicado por Amrami y Goldberg [2018], basado en un modelo de lenguaje bidireccional biLM ELMo [Peters et al., 2018] que resuelve un problema ligeramente más complejo que WSI. El problema en cuestión puede verse como WSI ponderado, asignando un valor de probabilidad a cada significado para una instancia dada.
En este trabajo, se enmarca el problema mediante algunas definiciones relevantes y una breve recorrida por lo que se consideró más destacable del estado actual de los problemas tanto de WSD como de WSI, para más adelante adentrarse en la implementación, entrenamiento y evaluación del modelo construido.

1.2. Objetivos y aportes

El objetivo principal de este trabajo es obtener y comparar resultados de WSD y WSI de la aplicación de una técnica de avanzada desarrollada para WSI en inglés aplicada al español

[2] Si bien para cada instancia la probabilidad se distribuye entre todos los clusters formados, para algunos de ellos (como el que corresponde a "banco de sangre" en el ejemplo) su valor puede ser 0 cuando no se considera aplicable para la instancia.

sobre una tarea internacional como es Senseval 2 *Spanish Lexical Sample*. El método elegido para este fin es *Word Sense Induction with Neural biLM and Symmetric Patterns* de Amrami y Goldberg [2018]. El aporte sustantivo realizado consiste en reportar resultados para WSI en español, de los cuales no hay antecedentes disponibles (o por lo menos no se encuentran accesibles).
Dado que este método implica la utilización de un modelo de lenguaje bidireccional biLM ELMo [Peters et al., 2018] entrenado para el español, y que no se encuentran disponibles modelos entrenados completos[3] en este idioma, surge la necesidad de realizar el entrenamiento de este modelo. La obtención de este modelo entrenado para el español es entonces un objetivo intermedio del presente trabajo. Este modelo es por tanto un aporte novedoso, que por su utilidad para futuras aplicaciones ha sido disponibilizado publicamente (https://github.com/lastrodrigo/SpanishElmoFullWeights).

1.3. Resultados esperados

Los resultados obtenidos en cuanto a la resolución de WSI no pueden compararse directamente a otros trabajos en español por la ausencia de resultados publicados (sobre WSI en particular), pero servirán como referencia para comparar contra trabajos futuros y los resultados disponibles en idioma inglés[4], y se evaluará también el rendimiento de la técnica en la realización de una de las tareas de WSD de SemEval disponibles para nuestro idioma (Senseval-2 *Spanish Lexical Sample Task*).
Además de los resultados obtenidos por la aplicación del método de WSI, un aporte importante de este trabajo es una red neuronal biLM ELMo entrenada en español, incluyendo el *dump* completo de sus pesos. Se trata de un recurso costoso de construir (su entrenamiento insumió unos 3 meses) que podrá ser reutilizado por futuros trabajos y que no se encontraba disponible anteriormente.

1.4. Organización del documento

El presente documento se estructura de la siguiente manera:
En la sección 2, se presentan conceptos teóricos fundamentales para enmarcar el problema y comprender el funcionamiento del método de desambiguación elegido. Asimismo, en esta sección se realiza una breve presentación de los antecedentes en el problema de WSD en español, principalmente a través de las tareas Senseval y SemEval realizadas para este idioma.
En la sección 3, se detalla la implementación llevada a cabo para el desarrollo de este trabajo, incluyendo el preprocesamiento del corpus para el entrenamiento del modelo de lenguaje, detalles del entrenamiento propiamente dicho, y las modificaciones necesarias en el desambiguador.

[3] Existe al menos un modelo biLM ELMo entrenado en español disponibilizado por Che et al. [2018], pero no cuenta con los pesos de la capa softmax, por lo que no puede ser utilizado como modelo de lenguaje.
[4] La comparación contra los resultados en inglés debe ser considerada tomando en cuenta las salvedades de que se trata de resultados para idiomas distintos, obtenidos sobre tareas distintas, etc.

En la sección 4 se presentan los resultados obtenidos para distintas variantes del método en la tarea Senseval 2 *Spanish Lexical Sample*, utilizando las métricas aplicadas en dicha tarea, así como las métricas formuladas para la tarea de WSI en inglés SemEval 2013 *Task* 13, tarea para la cual fue formulado originalmente el método estudiado.
Por último, en la sección 5 se presentan las conclusiones derivadas del trabajo realizado.

2. Marco teórico

En esta sección se presentan brevemente algunos conceptos básicos sobre los que se hará referencia a lo largo de este trabajo. Además de algunos conceptos teóricos del área de PLN y de inteligencia artificial (IA) en general, se presentan también las competencias internacionales SemEval y Senseval, por su importancia como fuente de recursos para el desarrollo de técnicas de PLN, así como marco de evaluación y comparación de las mismas.

2.1. Procesamiento de Lenguaje Natural

El procesamiento de lenguaje natural (PLN, o por sus siglas en inglés: NLP) es una disciplina que se sitúa en la intersección de otras disciplinas científicas como las ciencias de la computación, la inteligencia artificial, la lingüística y la psicología cognitiva [Kurdi, 2016]. Cuando se trata de modelos o formalismos desarrollados para ser implementados como programas de computadora, también se suele emplear el término “lingüística computacional” (*computational linguistics*, en inglés). En particular, el PLN se relaciona fuertemente con varias áreas de la inteligencia artificial, como la representación del conocimiento, la planificación, la percepción y el aprendizaje.
Podría decirse que el PLN se centra en analizar como una computadora puede “comprender” y “generar” el lenguaje humano, para lo cual suele dividirse el procesamiento del lenguaje natural en varias etapas, cada una con sus problemas. Así dentro del análisis sintáctico encontramos problemas como la lematización, *parsing* (determinar un árbol que refleje la relación entre las palabras en la oración), entre otros. Dentro del análisis semántico algunos problemas clásicos son la traducción automática, el reconocimiento de entidades, el análisis de sentimiento, y en particular WSD y WSI. El análisis del discurso trata sobre unidades lingüísticas mayores a la oración, como resúmenes automáticos, resolución de correferencias, etc. Dentro del análisis del habla, se tratan problemas como el reconocimiento del habla, la segmentación del discurso y la generación de habla a partir de texto. En la fig. 1, puede verse una representación del *pipeline* clásico de PLN con ejemplos de las tareas que pertenecen a cada etapa.

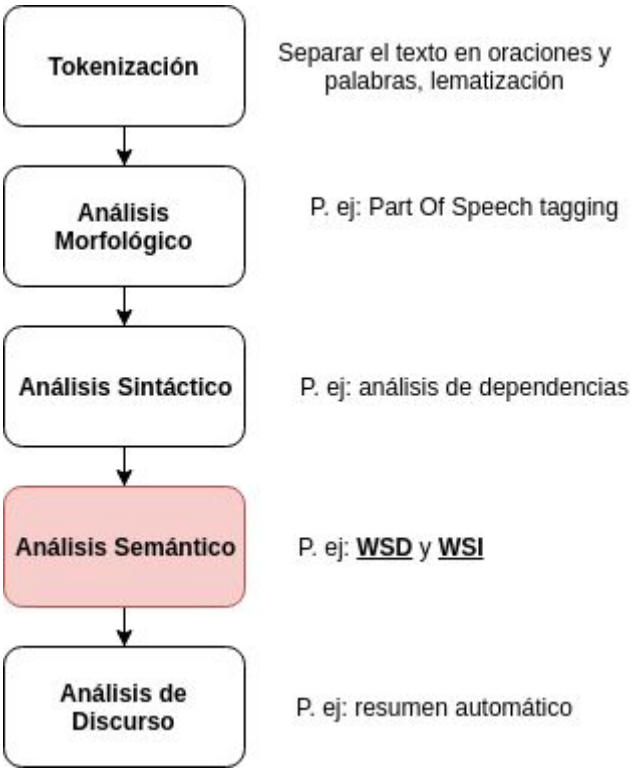

Fig. 1: Pipeline clásico de PLN con ejemplos de las tareas pertenecientes a cada etapa, donde se resalta WSD y WSI, las tareas sobre las que se profundiza en ese trabajo[5].

Desde la década de los 2000 [Kurdi, 2016], el aprendizaje automático ha ganado terreno en el PLN, dado que permite construir sistemas "inteligentes" con un esfuerzo menor que el necesario para construir sistemas basados en reglas, que requieren mayor carga de trabajo de expertos humanos. Sin embargo, su grado de penetración varía, desde casi un dominio total en el reconocimiento del habla, hasta un uso más limitado a mayores niveles de procesamiento como análisis del discurso, donde el paradigma simbólico es aún dominante. Hoy en día, a pesar de que el PLN es ampliamente utilizado por la industria, esto no ha disminuido el interés por las bases del procesamiento de lenguaje.

2.1.1. Modelos de lenguaje

En esta sección se presenta el concepto de modelo de lenguaje. Además de ser uno de los conceptos centrales de PLN, resulta de particular relevancia para el método de desambiguación que se utilizará en este trabajo.

[5] Adapatado de Garousi et al. [2018]

2.1.1.1. Definición y n-gramas

Un modelo de lenguaje es una distribución de probabilidad sobre secuencias de palabras. Para cada secuencia de palabras de largo m, un modelo de lenguaje asigna una probabilidad $P(w_1,...,w_m)$ a la secuencia.

Uno de los principales problemas a la hora de construir un modelo de lenguaje es la dispersión de los datos, lo que hace que muchas secuencias posibles de palabras no se encuentren en los datos de entrenamiento. Una solución posible pasa por asumir que la probabilidad de una palabra solo depende de las *n* palabras anteriores.

Bajo esta suposición, se tiene lo que se conoce como modelos de n-gramas. En su caso más simple, cuando *n=1* (conocido como unigrama), se descompone la probabilidad de la secuencia considerando que la probabilidad de aparición de una palabra en la misma es independiente de las palabras que la anteceden, de forma que

$$P(w_1,...,w_m)=P(w_1)P(w_2|w_1)...P(w_m|w_{m-1}...w_1) \approx \prod_{i=1}^{m} P(w_i)\,.$$

De forma más general, en el modelo de n-grama, la probabilidad de la oración $w_1,...,w_m$ se aproxima de la siguiente manera:

$$P(w_1, ..., w_m) = \prod_{i=1}^{m} P(w_i|w_1, ..., w_{i-1}) \approx \prod_{i=1}^{m} P(w_i|w_{i-(n-1)}, ..., w_{i-1})$$

Así se asume que la probabilidad de observar la i-ésima palabra w_i en el contexto de las *i-1* palabras precedentes se puede aproximar por la probabilidad de observarla en un contexto reducido de las *n-1* palabras anteriores. La probabilidad condicional puede calcularse a partir de conteos de frecuencia de n-gramas:

$$P(w_i|w_{i-(n-1)}, ..., w_{i-1}) = \frac{\#apariciones(w_{i-(n-1)},...,w_{i-1},w_i)}{\#apariciones(w_{i-(n-1)},...,w_{i-1})}$$

Sin embargo, por lo general las probabilidades de los modelos de n-gramas no se derivan directamente de conteos de frecuencia, dado que los modelos obtenidos de esta forma presentan problemas al enfrentarse a n-gramas que no han sido observados durante la construcción del modelo. Para ello se requieren técnicas conocidas como suavizado (*smoothing*), para asignar alguna parte de la probabilidad total a palabras o n-gramas no observados.

2.1.1.2. Modelos de lenguaje basados en redes neuronales

En las últimas décadas, propiciados por el crecimiento de la capacidad de cómputo y la accesibilidad de grandes volúmenes de datos en internet han surgido modelos de lenguaje basados en redes neuronales [Bengio et al., 2003]. Estos modelos utilizan representaciones en espacios vectoriales continuos para las palabras (habitualmente llamadas *word embeddings*, o simplemente *embeddings*) para realizar sus predicciones de probabilidad.

Los ***embeddings*** son representaciones de palabras como vectores, obtenidos utilizando técnicas de entrenamiento inspiradas en la hipótesis distribucional, es decir que palabras con significados similares tienden a ocurrir en contextos similares [Firth, 1961]. Estas representaciones han mostrado un buen rendimiento en una amplia variedad de tareas de PLN.

Los embeddings son una forma de atacar el problema de escasez de datos (*data sparsity problem*) derivado del entrenamiento de modelos de lenguajes entrenados en textos de cada vez mayor extensión, donde el tamaño del vocabulario aumenta, lo que a su vez conlleva un aumento exponencial del número de secuencias posibles. En general, dado un vocabulario *V*, los modelos neuronales se entrenan como clasificadores que buscan predecir la distribución de probabilidad sobre *V* dado un contexto lingüístico específico, $P(w_t|contexto)\ \forall w_t \in V$. Mikolov et al. [2013] propusieron dos modelos donde los embeddings son aprendidos usando una red neuronal *feed forward* de una única capa oculta, utilizando menos recursos computacionales que una RNN: el modelo CBOW (*Continuous Bag-of-Words*), que busca predecir una palabra basada en su contexto y el modelo *Skipgram* que intenta maximizar la clasificación de una palabra basada en otra palabra en la misma oración. En sus resultados muestran cómo al entrenar embeddings de dimensión alta en grandes cantidades de datos, es posible mediante operaciones básicas entre los vectores resultantes ver reflejadas relaciones semánticas entre las palabras, como por ejemplo entre un país y su capital, o como cuando al vector de la palabra *"rey"* se le resta el vector de la palabra *"hombre"* y se le suma el vector de la palabra *"mujer"* el vector resultante es muy próximo al vector de la palabra *"reina"*.

Otros modelos se basan en redes neuronales recurrentes, más complejas que las utilizadas por Mikolov et al. [2013]. Para la aplicación de las representaciones obtenidas por estos modelos pueden diferenciarse dos estrategias diferentes. Por un lado el enfoque basado en *features*, entre los cuales destaca la utilización de ELMO biLM[6], que será presentado en la próxima sección. Estas técnicas utilizan arquitecturas específicas que incluyen las representaciones pre entrenadas como *features*. Por otra parte, el enfoque basado en *fine-tunning* ("ajuste"), utiliza una mínima cantidad de parámetros específicos de la tarea y se entrena realizando pequeñas modificaciones a los parámetros pre entrenados. Dentro de estos, se encuentra BERT (*Bidirectional Encoder Representations from Transformers*) [Devlin et al., 2018]. Este modelo se diferencia de enfoques previos, limitados por tratarse de modelos de lenguaje unidireccionales , al pre entrenar representaciones bidireccionales profundas condicionadas conjuntamente en todas las capas por los contextos izquierdo y derecho. Esto se implementa con un nuevo modelo en el pre entrenamiento llamado "*masked language model*" (MLM), en el que se enmascaran algunas de las palabras de la entrada y el objetivo es predecir de qué palabra se trata únicamente por su contexto. Con la aplicación de este reciente modelo se obtuvieron nuevos estados del arte en varias tareas de PLN, tanto a nivel de oración como de palabra, superando a sistemas con arquitecturas específicas para las mismas. Los autores remarcan como algo clave para el éxito obtenido con este modelo al enfoque bidireccional para la representación del lenguaje, incluso más allá de la simple concatenación de representaciones de izquierda a derecha y de derecha a izquierda realizada por ELMo.

En el presente trabajo un modelo de lenguaje basado en redes neuronales se utilizará directamente para predecir palabras en un contexto, aunque cabe destacar que en muchas

[6] Este será el modelo utilizado en el presente trabajo, aunque no para utilizar los embeddings directamente como *features* de un sistema específico para la tarea, sino utilizando el modelo de lenguaje para obtener predicciones de palabras en un contexto.

ocasiones en lugar de utilizar los modelos de lenguaje basados en redes neuronales de esta forma, los embeddings obtenidos de los estados de las capas ocultas de la red neuronal son utilizados directamente.

ELMo biLM

El modelo de lenguaje ELMo biLM [Peters et al., 2018] consiste en dos modelos de lenguaje entrenados separadamente, un modelo "*forward*" entrenado para predecir $p\rightarrow(w_i|w_1,...,w_{i-1})$, y un modelo "*backward*" para predecir $p\leftarrow(w_i|w_n,...,w_{i+1})$. En adelante se utilizará la notación, sugerida por los autores, $LM\rightarrow(w_1w_2...w_{i-1}\_)$ para denotar la distribución $p\rightarrow(w_i|w_1,...,w_{i-1})$ y $LM\leftarrow(\_w_{i+1}w_{i+2}...w_n)$ para denotar $p\leftarrow(w_i|w_n,...,w_{i+1})$.

Este modelo de lenguaje basado en una red neuronal recurrente LSTM cuya arquitectura se presenta en la Fig. 2, introduce un nuevo tipo de representación que modela características complejas de la utilización de las palabras (sintácticas y semánticas) y como esos usos varían en distintos contextos lingüísticos (por ejemplo, para modelar la polisemia). Los vectores obtenidos son una función aprendida de los estados internos de un modelo de lenguaje bidireccional (biLM) de aprendizaje profundo, pre entrenado en un corpus de gran extensión.

Los vectores de ELMo (*Embeddings from Language Models*), son una combinación lineal de los vectores de estados de las capas de la RNN para cada palabra de entrada y para cada tarea específica donde sean utilizados, lo que mejora el rendimiento con respecto a la utilización de la última capa de la LSTM.

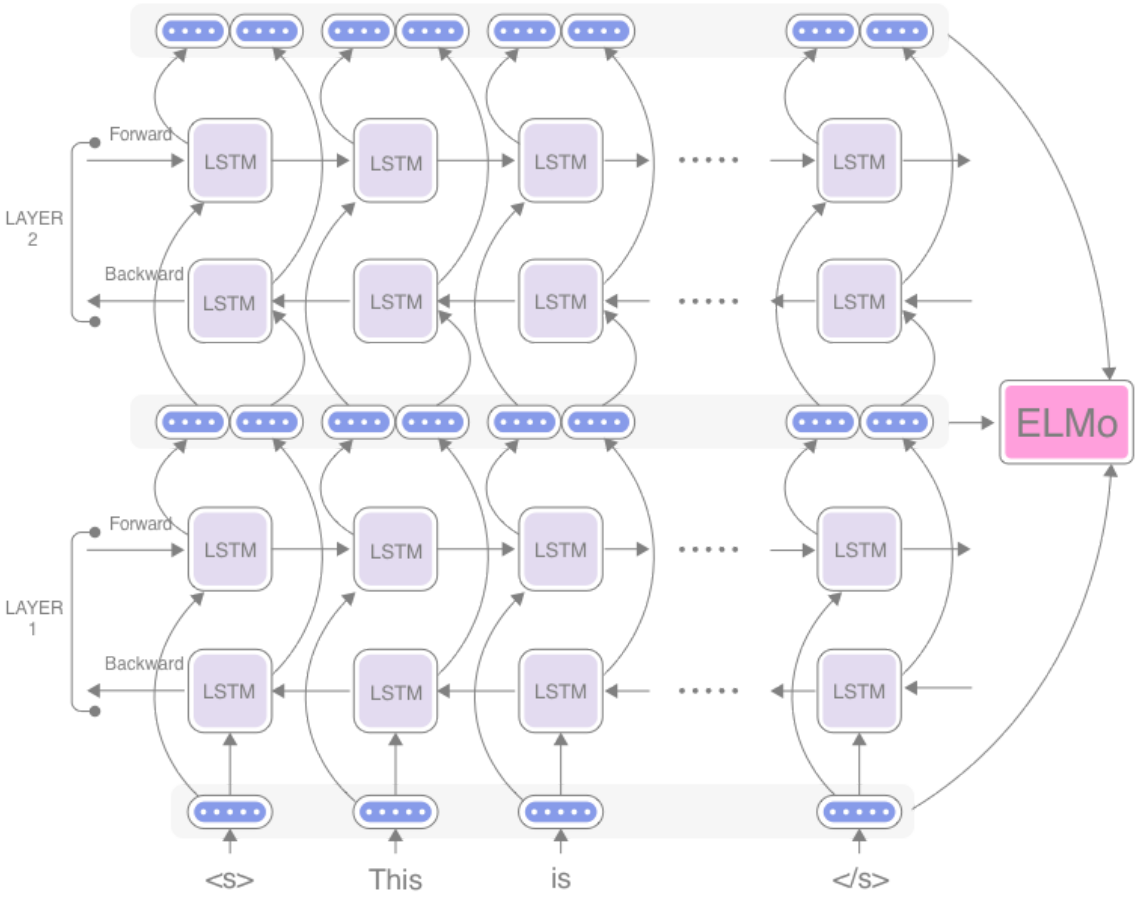

Fig.2: Esquema de biLM ELMo[7].

Las evaluaciones intrínsecas realizadas por sus creadores, mostraron que las capas superiores de la LSTM capturan aspectos dependientes del contexto del significado de las palabras (pueden usarse directamente para tareas de WSD), mientras que los estados de las capas inferiores modelan aspectos de la sintaxis (pueden usarse por ejemplo para etiquetado de POS). Así, los embeddings de ELMO pueden utilizarse para una amplia variedad de tareas de PLN. Los autores han reportado mejoras del estado del arte en diversos problemas con una disminución del error relativo de hasta un 20%.
El método de obtención de los embeddings también se beneficia de unidades subléxicas mediante el uso de convoluciones de caracteres, e incorpora información polisémica sin entrenar el modelo para predecir clases predefinidas.

2.1.1.3. Perplejidad de un modelo de lenguaje

La perplejidad es una medida utilizada en teoría de información para evaluar qué tan buena es una distribución de probabilidad para predecir una muestra de la variable aleatoria que modela.
Se define matemáticamente la perplejidad de una distribución *p* como:

[7] Tomada del blog Real World Natural Language Processing [Hagiwara, 2018].

$$2^{H(p)} = 2^{-\sum_x p(x) log_2 p(x)}$$

donde *H(p)* es la entropía en bits de la distribución y *x* varía sobre los valores posibles de la variable aleatoria. La entropía es una medida del número esperado de bits requeridos para codificar el valor de la variable aleatoria.
En el caso particular de una distribución aleatoria uniforme sobre *k* valores discretos, la perplejidad es igual a *k*.
En el contexto de PLN, dado que un modelo de lenguaje no es otra cosa que una distribución de probabilidad, se utiliza a la perplejidad como medida para evaluar modelos de lenguaje. Una forma de interpretar el resultado de la perplejidad en este contexto, es que dada una perplejidad por palabra *r* (lo que equivale a decir que las posibles palabras predichas por el modelo pueden codificarse utilizando $log_2(r)$ bits) el modelo está tan "*perplejo*" ante los datos utilizados para medir la perplejidad como si tuviera que elegir uniforme e independientemente entre *r* posibilidades para cada palabra.

2.1.2. WordNet y BabelNet

2.1.2.1. WordNet

WordNet es un sistema de referencia en línea diseñado según teorías psicolingüísticas de la memoria léxica humana. En él, sustantivos, verbos y adjetivos se organizan en conjuntos de sinónimos (*synonym sets*, o *synsets*), cada uno representando un concepto léxico subyacente. Dichos *synsets* se vinculan mediante diferentes relaciones léxicas [Miller et al, 1990].
En primer lugar, la relación de hiponimia genera una organización semántica jerárquica que se representa en WordNet mediante punteros etiquetados entre *synsets*, implementando un sistema de herencia léxica [Miller, 1990]. Así por ejemplo, el *synset* para "*árbol*" sería:

{árbol, @planta, ~conífera, ~aliso,...}

Donde "@" denota un puntero hacia un concepto que es hiperónimo, y "~" denota un puntero hacia un concepto hipónimo.
Otra relación modelada en WordNet es la meronimia, esto es la relación entre una parte y el todo. WordNet modela relaciones de la forma "*a* es un componente de *b*" (como la que existe entre "*rama*" y "*árbol*"), "*a* es un miembro de *b*" (por ejemplo, entre *"árbol"* y *"bosque*") o "*a* está hecho de *b*" (como entre *"mesa*" y *"madera"*).
La antonimia también se modela en WordNet, consideremos el caso de los *synsets* para "*hombre"* y *"mujer"*:

{[hombre, !mujer], @persona,...}
{[mujer, !hombre], @persona,...}

Donde "!" denota un puntero hacia un concepto que es antónimo, y los paréntesis rectos indican que la antonimia es una relación léxica entre las palabras, en lugar de una relación semántica entre conceptos.
Así cuando se tienen los distintos tipos de relaciones semánticas, se tiene una red de conceptos interconectados, como la que se observa en la figura 3.

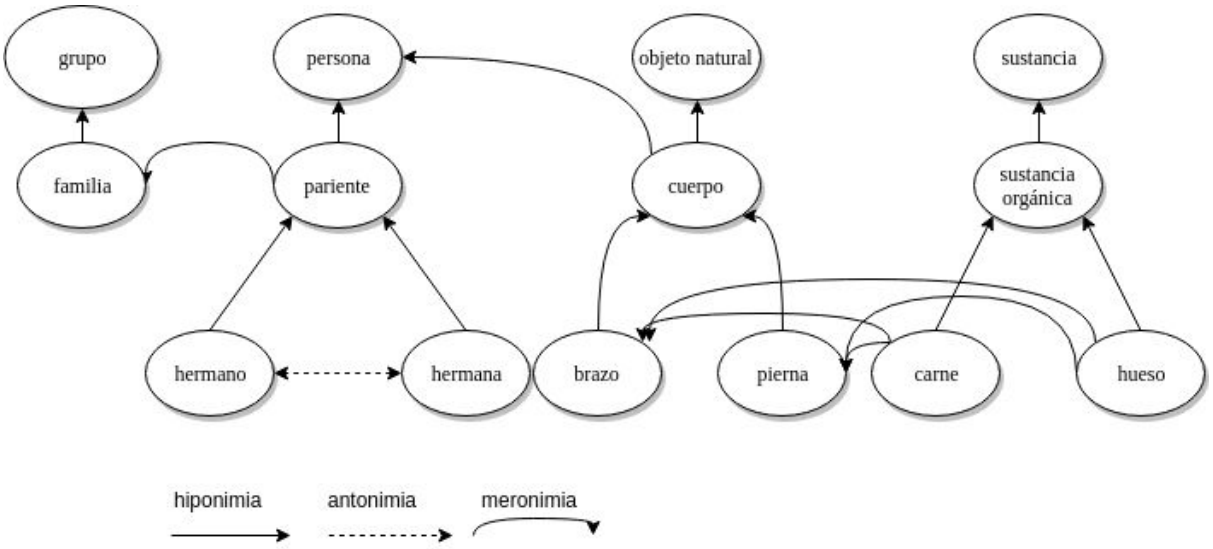

Fig. 3: Red de *synsets* interconectados mediante relaciones léxicas, adaptado de Miller [1990]

WordNet divide a las palabras en categorías: sustantivos, verbos, adjetivos, adverbios y palabras de función. El precio de imponer esta categorización sintáctica es una cierta cantidad de redundancia que los diccionarios convencionales evitan, ya que en WordNet una misma palabra puede aparecer en más de una categoría. Como contraparte, se tiene una clara visualización de las diferencias fundamentales en la organización semántica de las categorías, la cual puede ser explotada computacionalmente. Es allí donde radica la característica más ambiciosa (y útil) de WordNet, el organizar información léxica en términos de significados de las palabras, en lugar de por su forma. Por lo anterior es que WordNet puede verse como más similar a un tesauro que a un diccionario convencional.

2.1.2.2. BabelNet

BabelNet [Navigli y Ponzetto, 2010] es un recurso construido automáticamente mediante una metodología que integra conocimiento lexicográfico de WordNet con conocimiento enciclopédico de Wikipedia, a la vez que mediante traducción automática se enriquece el recurso asociando información léxica para múltiples lenguajes. Esta metodología surge como solución al problema de la construcción manual de este tipo de recursos lingüísticos estructurados para idiomas distintos al inglés en los cuales existe menos investigación y recursos que puedan utilizarse como materia prima. El resultado es una especie de diccionario enciclopédico que provee conceptos y entidades para varios idiomas conectadas mediante grandes cantidades de relaciones semánticas.

Puede verse el conocimiento contenido en BabelNet como un grafo dirigido *G=(V,E)* donde *V* es el conjunto de vértices (que representan los conceptos) y $E \subseteq V \times R \times V$ es el conjunto de aristas que conectan pares de conceptos relacionados semánticamente, donde *R* es el conjunto de etiquetas que determinan el tipo de relación. Cada vértice en *V* contiene un conjunto de traducciones para distintos idiomas. Tomando WordNet, se toman los significados como conceptos y los vínculos semánticos entre *synsets* como relaciones,

luego se toman las entradas de Wikipedia también como conceptos y se consideran relaciones semánticas sin especificar a partir de los hipervínculos entre entradas. Para unificar conceptos repetidos se unen aquellos que figuran tanto en Wikipedia como en WordNet.
Para incorporar la información en múltiples idiomas, se obtienen las palabras equivalentes al concepto en cada lenguaje a partir de los vínculos entre versiones de distintos idiomas de una misma entrada de Wikipedia y un sistema de traducción automática.
Un *synset* de BabelNet, cuenta con las mismas relaciones léxicas etiquetadas que su contraparte en WordNet, más relaciones léxicas extraídas de los artículos relacionados a su contraparte en Wikipedia, más términos equivalentes en otros idiomas extraídos de las versiones para otros lenguajes de la entrada de Wikipedia y de la aplicación de traducción automática de instancias extraídas de Wikipedia y el corpus SemCor, como puede verse en la figura 4, para el ejemplo de la palabra en inglés *"balloon*" en su sentido equivalente al concepto en español *"*globo aerostático":

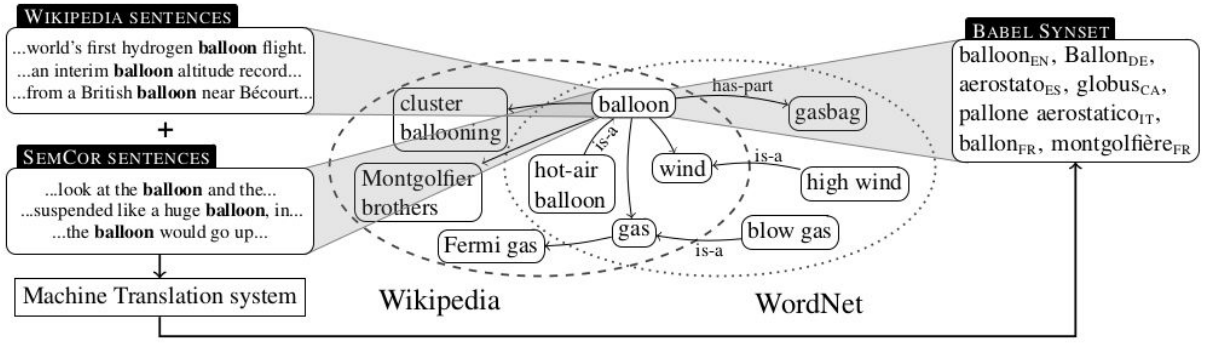

Fig. 4: Relación entre un *synset* de BabelNet y sus contrapartes en WordNet y Wikipedia, extraído de [Navigli y Ponzetto, 2010].

2.2. Aprendizaje automático

El aprendizaje automático (en inglés *Machine Learning*, abreviado ML), puede considerarse como un área de la inteligencia artificial que estudia los algoritmos, heurísticas y modelos estadísticos que las computadoras pueden utilizar para realizar tareas de forma eficaz basándose en patrones e inferencias, en lugar de instrucciones específicas [Russell y Norvig, 2016].
Para resolver un problema en una computadora se necesita un algoritmo, es decir, una secuencia ordenada de instrucciones que puedan llevarse a cabo para transformar la entrada en la salida . Para algunas tareas no es sencillo encontrar un algoritmo, ya que no está claro cómo transformar la entrada en la salida. Sin embargo, para algunas de estas tareas existen grandes cantidades de datos disponibles para analizar.
En estos casos, quisiéramos que la computadora extrajera automáticamente el algoritmo para llevar a cabo la tarea a partir de los datos [Alpaydin, 2009]. El avance de la tecnología nos permite almacenar y procesar estos enormes volúmenes de datos, así como accederlos desde cualquier parte del mundo a través de redes de computadora. Pero esto solo se

vuelve útil cuando esos datos son analizados y convertidos para su utilización, por ejemplo, para hacer predicciones.
Aunque no conozcamos los detalles de los procesos (que pueden ser biológicos, físicos, o de cualquier otra naturaleza) que generan esos datos, sabemos que no son completamente aleatorios. Existen patrones en los datos. Puede que entonces no seamos capaces de identificar el proceso completamente, pero sí podemos construir una aproximación lo suficientemente útil.
Estos patrones pueden ayudarnos a entender el proceso, y más aún: a hacer predicciones. Asumiendo que el futuro, por lo menos cercano, no será muy distinto que el pasado (por lo menos el reciente) cuando los datos se obtuvieron, cabe esperar que las predicciones sean correctas.
Los algoritmos de ML ajustan un modelo matemático basándose en datos de muestra, proceso que se denomina “entrenamiento”. A grandes rasgos, suele clasificarse el mecanismo de aprendizaje como supervisado, no supervisado o por refuerzo:

- **Aprendizaje supervisado:** el agente observa ejemplos consistentes en la entrada y la salida esperada y aprende la función que mapea la entrada con la salida.
- **Aprendizaje no supervisado:** el agente aprende patrones en la entrada, sin que se le proporcione retroalimentación sobre sus predicciones.
- **Aprendizaje por refuerzo:** el agente aprende una serie de refuerzos en forma de recompensas y castigos.

Luego se aplica el modelo para hacer predicciones o tomar decisiones sobre otros conjuntos de datos que sean de interés mediante reconocimiento de patrones e inferencias en base a lo aprendido de los datos observados en la etapa de entrenamiento.
Esta idea es aplicable a las más diversas áreas, desde el comercio minorista, o el análisis crediticio, hasta el análisis de grandes volúmenes de datos físicos, astronómicos o biológicos, pasando por patrones de llamadas para la optimización de redes de telecomunicaciones. También puede utilizarse para el análisis del lenguaje, en particular para la desambiguación de palabras, una tarea que los humanos realizamos con facilidad (al menos, en la mayoría de los casos). Somos capaces de reconocer el significado que se le da a una cierta palabra en un determinado contexto, aún cuando nunca hayamos escuchado o leído esa misma oración o texto. De la misma manera, analizando múltiples oraciones donde se utiliza una cierta palabra con un determinado significado, un programa de computadora captura los patrones específicos del contexto y es capaz de reconocer con qué significado se utiliza la palabra en una oración dada.
El aprendizaje automático combina teorías estadísticas para construir modelos matemáticos, con ciencias de la computación, para utilizar algoritmos eficientes para la fase de optimización y de inferencia.

2.2.1. Redes neuronales

Un método de aprendizaje automático que ha ganado popularidad en los últimos años son las redes neuronales. Una red neuronal es un conjunto de elementos de procesamiento interconectados, llamados unidades o nodos, cuya funcionalidad se inspiró inicialmente en las neuronas biológicas [Gurney, 2014]. La habilidad de procesamiento de la red se

encuentra en la “intensidad” (llamada “peso”) de las interconexiones entre estas unidades, los cuales surgen de un proceso de adaptación (entrenamiento) a partir de un conjunto de ejemplos de entrenamiento.

En el mundo animal, las neuronas se comunican mediante señales eléctricas o químicas que conllevan breves impulsos o picos de voltaje en la membrana de la célula. A través de las sinapsis, estas señales eventualmente llegan al cuerpo de otra célula, donde son integradas o sumadas de alguna manera y, a grandes rasgos, si la señal resultante excede un cierto umbral la neurona emite o genera un nuevo impulso eléctrico como respuesta. Para determinar si un impulso debe producirse o no, algunas señales entrantes a la neurona producen un efecto inhibitorio y tienden a prevenir la generación del impulso, mientras otras producen un efecto excitatorio que promueve la generación del mismo. Esta distinción en la forma de procesamiento de cada neurona se supone que reside en el tipo (excitatorio o inhibitorio) y la fuerza de sus conexiones sinápticas con otras neuronas.

Es esta estructura y lógica de procesamiento la que históricamente se ha buscado incorporar en las redes neuronales artificiales, y por el énfasis en la importancia de las interconexiones entre neuronas a este tipo de sistemas en ocasiones se les denomina “conexionistas”.

En las redes neuronales artificiales, el equivalente a las neuronas son los nodos ya mencionados al comienzo, y cada sinapsis se modela mediante un número conocido como peso, por el que se multiplica cada entrada antes de llegar a la neurona, donde estas señales de entrada se suman para obtener la *activación* de la neurona. En el caso de la neurona artificial más simple (llamada *TLU*, por *Threshold Logic Unit*), la cual se muestra en la figura 5, la activación se compara con un valor umbral. En caso que la activación sea mayor que dicho umbral el nodo produce un valor de salida “activado” (habitualmente 1), en otro caso la salida es “inactiva” (habitualmente 0).

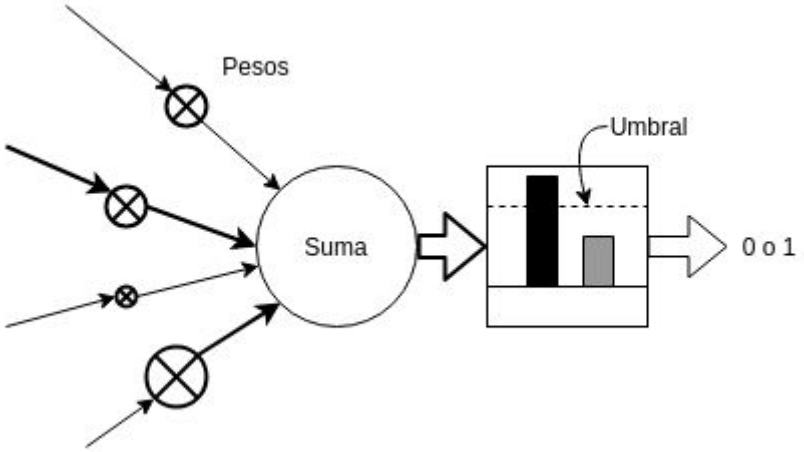

Fig. 5: Neurona artificial simple[8].

[8] Basado en [Gurney, 2014]

Generalizando el modelo de neurona artificial, tenemos que se trata de un nodo que provee una combinación lineal de *N* pesos $w_1,...,w_N$ y *N* entradas $x_1,...,x_N$, y aplica una función no lineal *Φ.* Matemáticamente, el término neurona puede definirse como un operador que realiza una transformación $R^{N+1} \rightarrow R$, cuya ecuación general es:

$$y = \Phi(\sum_{i=1}^{N} w_i x_i + w_0)$$

Donde w_0 es un peso que corresponde a un sesgo definido para la neurona y la función $\Phi: R \rightarrow R$ es monótona y continua (además, es típicamente una función sigmoide) [Mandic y Chambers, 2001].

Los modelos de redes neuronales se definen en base a las características de las neuronas utilizadas, las reglas de aprendizaje y la topología de la red. El diseño topológico más básico consiste en ordenar los nodos en capas en la cual las señales de entrada llegan a todos los nodos de la primera capa, y luego la salida de estos es tomada como entrada por la segunda capa, y así sucesivamente hasta que se alcanza la capa de salida donde ya no se realizan más transformaciones. Este tipo de redes neuronales se conoce como *feedforward* (Fig. 6). En estas topologías no existen ciclos en el grafo de la red. Otra categoría de topologías de red neuronal son las redes recurrentes (Fig. 7), las cuales si cuentan con ciclos en su topología debido a que cuentan con conexiones llamadas de *feedback*.

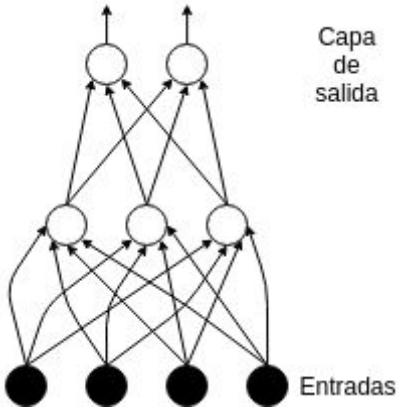

Fig.6: Respresentación de una red neuronal *Feed Forward*[9].

[9] Ibíd.

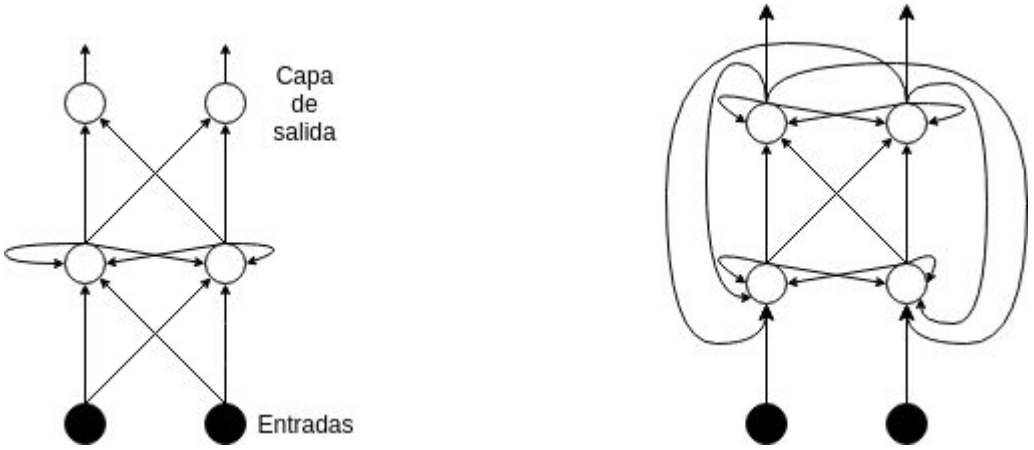

Fig. 7: a la izq. Red con una capa de entrada, una capa oculta completamente recurrente y una capa de salida, a la derecha red recurrente completamente conectada.

Dentro de las redes recurrentes, se encuentran las denominadas redes LSTM (de *Long Short Term Memory*), en las cuales las unidades (neuronas) típicas son reemplazadas por bloques de memoria (*memory blocks*). Un bloque de memoria contiene una o más celdas de memoria (*memory cells*), cada una conteniendo una neurona "típica" (que en este contexto se denomina CEC, *Constant Error Carousel*). Por cada bloque existen además dos compuertas (*gates*) adaptativas, una de entrada y otra de salida, que permiten o evitan el pasaje de las entradas o salidas a todas las celdas del bloque [Gers, 2001]. La red neuronal que se utilizará en el presente trabajo, sobre la cual se profundizará más adelante, será una red LSTM.
El "conocimiento" que contiene la red se encuentra en los pesos, los cuales se ajustan como se ha mencionado anteriormente, durante el proceso de entrenamiento.

2.3. Word Sense Disambiguation (WSD) y Word Sense Induction (WSI)

En esta sección se profundiza sobre los problemas de WSD y WSI, se presenta un resumen de las diferentes tareas de WSD en español y se introducen algunas técnicas de WSI: desde un primer enfoque sencillo, pasando por AI-KU, antecedente inmediato de WSI *with neural biLM and Symmetric Patterns*, para finalmente llegar a este último que será el método a aplicar sobre la tarea Senseval 2- Spanish Lexical Sample.

2.3.1. Formalización del problema

Si se ignora los signos de puntuación, puede verse un texto *T* como una secuencia de palabras $(w_1,w_2,...,w_n)$, y describir WSD como la tarea de asignarle el significado adecuado a

todas o algunas de las palabras en T[10]. Es decir, identificar un mapeo A de palabras a sentidos, tal que $A(i) \subseteq Significados_D(w_i)$ donde $Significados_D(w_i)$ es el conjunto de los sentidos contenidos en un diccionario D para la palabra w_i, donde $A(i)$ es el subconjunto de significados de w_i apropiados en el contexto de T. El mapeo A puede asignar más de un sentido a cada palabra $w_i \in T$, aunque típicamente sólo se toma el significado que se considere como de mayor probabilidad [Navigli, 2009].

Como ya se ha mencionado, puede verse WSD como un problema de clasificación, donde los significados de la palabra son las clases y mediante un método de clasificación automática se asigna a cada ocurrencia una o más clases basándose en el contexto de la ocurrencia y en fuentes de conocimiento (tesauros, ontologías, etc.).

Consideremos por ejemplo la oración "Depositó el dinero en el banco". Si se quiere aplicar WSD para la oración, tendremos *T=(depositó, el, dinero, en, el, banco)*, donde podemos querer desambiguar todas o algunas de esas palabras. Tomando como ejemplo la palabra *"banco"* (la sexta palabra en la oración) y suponiendo que nuestro inventario de significados D es el diccionario de la RAE, el objetivo es encontrar $A(6) \subseteq Significados_{RAE}(banco)$, que es el conjunto de los significados más apropiados en el diccionario de la RAE para esa instancia de la palabra "*banco*".

En el caso de WSI, la diferencia consiste en que en lugar de $Significados_D(w_i)$, dado que no existe un diccionario D con significados predefinidos, se tiene un conjunto de $Significados_{T'}(w_i)$ inducidos a partir de un texto T' para la palabra w_i. Además en el problema planteado en la tarea SemEval 2013 *Task* 13, se trata de encontrar una distribución de probabilidad sobre $A(i)$.

En este caso, para la misma oración de ejemplo, se tiene un conjunto $Significados_{T'}(banco)$ inducidos a partir de un texto T' (típicamente un texto de gran extensión, por ejemplo todo el texto existente en Wikipedia), y se tiene $A(6) \subseteq Significados_{T'}(banco)$. En el caso de SemEval 2013 *Task* 13, el mapeo A asigna además una probabilidad a cada significado del conjunto de significados inducidos.

La figura 8 muestra a través de un ejemplo las diferencias entre WSD, WSI y WSI ponderado.

[10] Se distingue en este caso dos variantes de WSD, a saber: *lexical sample* en caso que se trate de algunas de las palabras, y *all words* si se desambiguan todas las palabras.

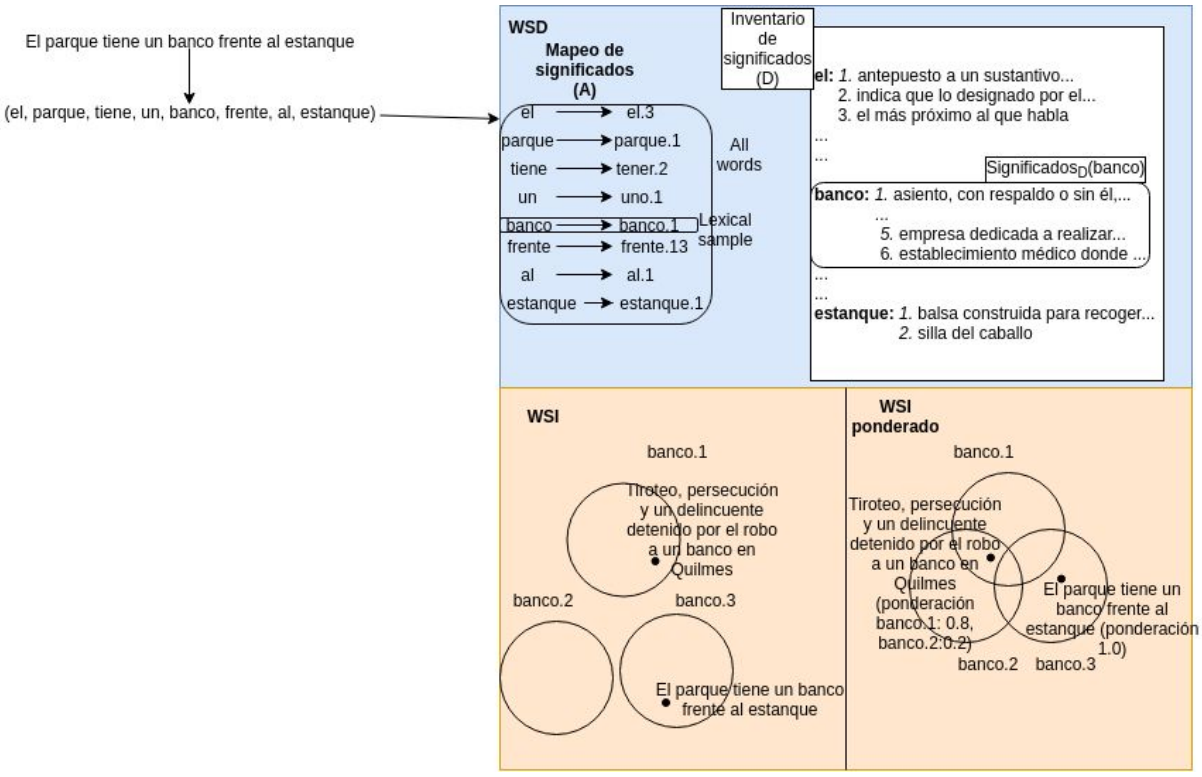

Fig. 8: Esquematización de WSD, WSI y WSI ponderado.

2.3.2. Word Sense Disambiguation (WSD)

Visto como un problema de aprendizaje automático, los posibles significados que provee el inventario son las etiquetas con las que se clasifican las instancias (ocurrencias) de una palabra dada.

Los principales enfoques utilizados tradicionalmente para la implementación de WSD pueden dividirse en las siguientes categorías:

- WSD Supervisado: son enfoques que usan métodos de aprendizaje automático supervisado para entrenar un clasificador a partir de un conjunto extenso de ejemplos anotados manualmente (es decir, desambiguados por un humano) a partir de un inventario de significados en particular.
- Métodos Basados en Conocimiento: son métodos que utilizan recursos lingüísticos como diccionarios, tesauros, ontologías, etc, para determinar el sentido de las palabras en su contexto, típicamente mediante técnicas basadas en grafos para descubrir y explorar propiedades del grafo subyacente a una o varias de estas bases de conocimiento lingüístico [Agirre, 2009].

2.3.2.1. Métricas para evaluación de WSD

Dado que WSD puede verse como un problema de clasificación, tradicionalmente se utilizan las mismas métricas que para otros problemas de este tipo: precisión, *recall* y F1.

$$Precisión = \frac{Verdaderos\ Positivos}{Verdaderos\ Positivos + Falsos\ Positivos}$$
$$Recall = \frac{Verdaderos\ Positivos}{Verdaderos\ Positivos + Falsos\ Negativos}$$
$$F1 = \frac{2\ x\ Precisión\ x\ Recall}{Precisión + Recall}$$

En el contexto de WSD, un verdadero positivo consiste en una instancia para la cual el sistema evaluado ha asignado una etiqueta y esta concuerda con el etiquetado considerado correcto (*gold standard*). Un falso positivo, es una instancia para la cual el sistema asignó una etiqueta, pero esta resulta ser distinta al etiquetado considerado correcto. Un falso negativo, es toda instancia para la que no se asignó su etiqueta correcta, ya sea porque el sistema asignó una etiqueta incorrecta o directamente no produjo una etiqueta para esa instancia.

2.3.2.2. WSD en Español

En esta sección se describen algunos hitos en la historia de WSD para el español, vinculados a la realización de tareas de WSD para este idioma en las evaluaciones internacionales Senseval y SemEval. SemEval (de Semantic Evaluation) es una serie de evaluaciones de sistemas de análisis semántico, como continuación de la serie de evaluaciones Senseval para el sentido de palabras. Estas evaluaciones buscan profundizar en la naturaleza del significado en el lenguaje.
En un comienzo (Senseval 1, 2 y 3), se trataba de tareas basadas en el sentido de las palabras (WSD). A partir del cuarto evento de la serie, SemEval 2007, las evaluaciones han avanzado hacia la investigación de áreas como las relaciones entre los elementos de una oración, relaciones entre oraciones y el análisis del sentimiento (*sentiment analysis*).
Las pruebas, que comenzaron para el idioma inglés, fueron incorporando más idiomas y aumentando la cantidad de equipos involucrados.
Cabe mencionar, que las 4 tareas que se presentan a continuación fueron consideradas en un principio como posibles evaluaciones para aplicar el método a evaluar en este trabajo. Lamentablemente, no fue posible encontrar los recursos asociados a la tarea para Senseval 3 *Spanish Lexical Sample*, por lo que fue descartada.
Finalmente se eligió Senseval 2 *Spanish Lexical Sample* por sobre SemEval 2013 Task 12 (que además consiste en anotar únicamente sustantivos) y SemEval 2015 Task 13 por ser estas últimas tareas *all-words* y la etapa de mapeo[11] de etiquetas a aplicar a la salida de WSI con biLM ELMO requiere de un número considerable de ejemplos para entrenar y no resultaba aplicable para un porcentaje significativo de las palabras en el corpus de estas tareas, por contar estas con pocas instancias en los corpus de test.

Senseval 2- Spanish Lexical Sample Task
A fin de contar con una tarea de referencia para la medición de los resultados obtenidos por el desambiguador para el idioma español se consideraron las presentadas en las competencias Senseval y SemEval hasta la fecha. Por su similaridad con la tarea *SemEval 2013 Task 13* en inglés se eligió la tarea de *Spanish Lexical Sample* de Senseval 2.

[11] Sobre este procedimiento de mapeo se profundizará en la sección 2.3.4.1.

Se trata de una tarea de WSD *lexical sample* (solo se intenta desambiguar el significado de un grupo seleccionado de palabras objetivo, de las cuales solo una de ellas aparece en cada oración) para 40 palabras (18 sustantivos, 13 verbos y 9 adjetivos). Las palabras pertenecen únicamente a una categoría sintáctica y las oraciones son elegidas para ilustrarlo.
La tarea provee un diccionario creado especialmente para la misma que incluye una definición para cada sentido, vinculada a la versión en español de EuroWordNet, y por lo tanto a WordNet 1.5 en inglés. También incluye la categoría sintáctica, y en algunos casos ejemplos y sinónimos. Las definiciones fueron construidas especialmente para la tarea, independientemente de WordNet en español.

Resultados
Los mejores resultados fueron alcanzados por variantes del sistema de JHU, llegando a alcanzar un 0.71 de precisión y recall, seguido del sistema CS224N de la Universidad de Stanford y el sistema SST.
El sistema presentado por JHU consiste en 6 subsistemas de aprendizaje supervisado integrados mediante un clasificador. Los subsistemas incluyen listas de decisión, aprendizaje basado en transformaciones, modelos vectoriales y sistemas basados en *naive* Bayes.
Más información sobre los sistemas participantes puede encontrarse en el anexo A.

Senseval 3- Spanish Lexical Sample
La tarea propuesta por Màrquez et al. [2004] fue diseñada para evaluar sistemas supervisados y semi supervisados de WSD, de forma coordinada con otras cinco tareas similares para distintos idiomas (vasco, catalán, inglés, italiano y rumano) para compartir unas 10 palabras objetivo. Considera 46 palabras, de las cuales 21 son sustantivos, 7 adjetivos y 18 verbos, manteniendo algunas palabras de la tarea Senseval-2 *Spanish Lexical Sample*. Para su realización se construyó un diccionario denominado MiniDir-2.1, con una polisemia promedio de 5.33 significados por palabra, 4.52 para los sustantivos, 6.78 para los verbos y 4.0 para los adjetivos.
Cada significado en MiniDir-2.1 está vinculado a su correspondiente en EuroWordNet y contiene información sintagmática (colocaciones, ejemplos, etc.) extraída de corpus. Con este diccionario se anotó el corpus MiniCors, consistente en 12625 instancias etiquetadas en 35875 oraciones con 1506233 palabras. El contexto considerado para cada instancia a desambiguar consiste en la oración que la contiene más la inmediatamente anterior y posterior.
Para cada palabra se tiene un mínimo de 200 ejemplos anotados manualmente por tres anotadores expertos independientes y los casos de desacuerdo fueron resueltos por un cuarto anotador asignando un único significado por ejemplo. A los participantes se les facilitó el diccionario MinDir-2.1, un conjunto de entrenamiento consistente en ⅔ de los ejemplos anotados, un conjunto de test formado por el tercio restante y un conjunto complementario de ejemplos sin anotar con un máximo de 1500 instancias por palabra.

Como ayuda a los participantes, las oraciones en todos los conjuntos de datos se presentaron tokenizadas, lematizadas y etiquetadas con POS.

Resultados
Siete equipos participaron, presentando 9 sistemas distintos (7 supervisados y 2 no supervisados). Los sistemas basados en *Support Vector Machines* y *Maximum Entropy* fueron los que obtuvieron mejores resultados, sin diferencias significativas entre ellos, con una precisión, recall y F1 del entorno de 0.84. Todos los sistemas supervisados superaron la línea base del significado más frecuente, hasta en un 16% en el caso de los sistemas con mejores resultados. En cambio, los sistemas no supervisados no lograron superar a la línea base del significado más frecuente.

SemEval 2013 Task 12: Multilingual Word Sense Disambiguation
La tarea [Navigli et al., 2013], consiste en que los sistemas participantes anoten sustantivos en un corpus de test con el significado más apropiado de BabelNet, WordNet o Wikipedia. El *gold standard* se elaboró anotando el corpus con BabelNet 1.1.1, lo que implica que se encuentra a su vez anotado en WordNet 3.0 y Wikipedia, dado que estos dos últimos recursos se encuentran vinculados a los conceptos en BabelNet. A diferencia de otras tareas de WSD *all-words*, en esta tarea no se toman en cuenta verbos, adjetivos ni adverbios dado que al momento de su realización estas categorías gramaticales no se encontraban cubiertas por BabelNet para idiomas distintos al inglés.
El corpus de test consta de 13 artículos de distintas áreas, que tratan desde deportes hasta noticias financieras. Cada artículo fue disponibilizado en 4 idiomas: inglés, francés, alemán y español. Dado que el *gold standard* fue anotado usando BabelNet, y que los conceptos presentes en este recurso se encuentran a su vez o bien incluídos en WordNet o en Wikipedia (o en ambos), se permitió a los sistemas participantes realizar sus anotaciones utilizando cualquiera de esos 3 inventarios de significados, tomándose en cuenta en el caso de las anotaciones realizadas en base a WordNet o Wikipedia únicamente aquellas instancias anotadas con significados de BabelNet relacionados con significados del recurso correspondiente. Para la evaluación se utilizaron las definiciones estándar de precisión, recall y F1.
Participaron de la tarea 3 equipos, con un total de 7 sistemas, cada equipo presentó al menos un intento para cada combinación de inventario de significados y lenguaje. Algo a destacar es que todos los sistemas presentados están basados en grafos, ya sea utilizando WordNet o BabelNet.

Resultados
El sistema UMCC-DLSI *Run-2* fue el que obtuvo los mejores resultados para todos los lenguajes evaluados. En español, obtuvo un valor de F1 de 0.710, seguido de *Run-1* que obtuvo un 0.705, DAEBAK con 0.600 y GETALP BN-2 con 0.371.
Cabe notar que *Run-2* toma ventaja de la heurística de “un sentido por discurso”, que utiliza la misma etiqueta de significado para todas las ocurrencias de un lema en un documento.

UMCC-DLSI presentó tres sistemas basados en ISR-WN de Gutiérrez et al. [2011] que enriquece la red semántica de WordNet utilizando aristas de varios recursos léxicos. Luego la desambiguación se realiza utilizando la red construida con el algoritmo presentado por Gutiérrez [2012] que extiende el algoritmo de WSD de *Personalized Page Rank* de Agirre y Soroa [2009] que incluye información sobre la frecuencia del significado. Sólo los sistemas *Run-1* y *Run-2* se presentaron para la evaluación en español. El sistema *Run-1* realiza WSD usando todas las instancias de sustantivos en el contexto de la oración, mientras que *Run-2* funciona a nivel de discurso e inicializa *PageRank* utilizando los *synsets* de todos los sustantivos en el documento.
Más información sobre los sistemas participantes puede encontrarse en el anexo A.

SemEval 2015 Task 13: Multilingual All-Words Sense Disambiguation and Entity Linking

Esta tarea, llevada adelante por Moro y Navigli [2015], busca evaluar en forma conjunta la resolución de dos problemas clásicos del Procesamiento de Lenguaje Natural: WSD y *Entity Linking* (por sus siglas, *EL*). *Entity Linking* es una tarea surgida más recientemente que WSD, que busca descubrir referencias a una entidad en un texto y vincularlas a la entrada más aplicable de una base de conocimiento. Aunque ambas tareas giran en torno al manejo de la ambigüedad léxica inherente al lenguaje, la principal diferencia pasa por el tipo de inventarios de significados que utilizan. Mientras que WSD requiere de información lexicográfica, EL utiliza conocimiento enciclopédico.
Los autores Moro y Navigli, buscaron en esta tarea analizar cómo el uso de un recurso que integrara ambos tipos de “conocimiento” (BabelNet) podía aplicarse para la resolución de WSD y EL mediante métodos similares, dejando abierta también la participación a sistemas que solo realizaran una de las tareas, e inclusive a sistemas de WSD que realizaran desambiguación exclusivamente de ciertas categorías gramaticales. Más aún, desarrollaron un conjunto de datos para la tarea en tres idiomas (inglés, italiano y español), con textos paralelos independientemente anotados manualmente por hablantes de cada idioma respectivamente. La tarea, consiste entonces en anotar 4 documentos tokenizados y etiquetados con categorías gramaticales (*POS-tagged*) que se proveen en cada uno de los 3 idiomas. Para la evaluación se utilizaron las métricas tradicionalmente utilizadas en el área: precisión, *recall* y F1.
A diferencia de tareas anteriores, en la tarea no se especifica a los participantes qué fragmentos del texto se debe desambiguar, a modo de crear un ambiente más realista así como de seguir la tendencia en otras tareas de EL. Dos de los documentos proporcionados contienen información médica sobre distintos fármacos, otro documento es el manual de un software que implementa una calculadora gráfica, y el restante contiene una discusión sobre asuntos sociales, especialmente vinculados con el apoyo para trabajadores de edad avanzada y a la problemática del desempleo en general llevada a cabo por la Comisión Europea. El inventario de significados utilizado es la red semántica multilingüe y diccionario enciclopédico BabelNet 2.5.1
El proceso de anotación manual de los documentos se realizó de forma paralela para cada lenguaje, con diferentes anotadores trabajando en cada lenguaje específico. Para cada

lenguaje, trabajaron 2 anotadores, más un tercero que actuó como juez, revisando todos los items y descartando anotaciones demasiado generales o irrelevantes y resolviendo el desacuerdo entre los anotadores. Como resultado se obtuvo un conjunto de datos con 1200 items, con 80 entidades mencionadas para cada lenguaje.
Para aquellos sistemas que dan como salida múltiples respuestas para una instancia, se pondera uniformemente las respuestas al computar la cantidad de verdaderos positivos. Como línea base se considera el primer sentido en BabelNet.
De los sistemas participantes, solo 2 resolvieron WSD para todas las categorías gramaticales para el idioma español: LIMSI y SUDOKU.

Resultados
En lo que respecta al idioma español, SUDOKU *Run 2* obtuvo los mejores *recall* y F1 (54.6 y 57.1) y SUDOKU *Run 1* la mejor precisión (60.2). Se observa una baja considerable en el rendimiento de LIMSI, que fuera el sistema con mejores resultados para el inglés, quedando por debajo de SUDOKU. El rendimiento de este último resulta interesante dado que obtiene los mejores rendimientos en español e italiano, a la vez que el segundo puesto en inglés donde participaron 6 sistemas distintos.
En términos generales, los rendimientos en español e italiano resultaron más bajos que en inglés, con SUDOKU confirmando su capacidad para explotar las palabras monosémicas arrojando métricas comparables a las obtenidas en inglés, mientras que LIMSI pierde casi un 20% en sus resultados. SUDOKU también obtuvo los mejores resultados para español e italiano para todas las categorías gramaticales, excepto los adverbios en italiano. Otro resultado interesante es que los métodos supervisados probaron ser difíciles de generalizar para múltiples idiomas, lo que produjo que los métodos supervisados que se presentaron a la tarea solamente lo hicieran para el inglés.
SUDOKU es un método no supervisado basado en *Personalized Page Rank* de Agirre et al. [2014], donde este se inicia con un vector sesgado hacia palabras monosémicas. Presenta 3 variantes: *Run 1* consta del método base propuesto por Manion y Sainudiin [2014] aplicado a nivel del documento, *Run 2* es la versión iterativa de la variante anterior aplicada a nivel de documento y con las palabras desambiguadas en orden creciente de polisemia, *Run3* es análogo a *Run 2*, pero primero aplicado primero a sustantivos, y luego a verbos, adjetivos y adverbios.
Más información sobre los sistemas participantes puede encontrarse en el anexo A.

2.3.3. Word Sense Induction (WSI)

A diferencia de WSD, el problema pasa de ser cómo elegir los significados más adecuados de un inventario predefinido a cómo descubrir los significados automáticamente a partir de texto limpio, sin marcas ni anotaciones de ningún tipo.
La forma de lograrlo consiste en agrupar en *clusters* (conglomeraciones o grupos) las ocurrencias de las palabras, basándose en la hipótesis distribucional [Firth,1961] . Esto es, una palabra dada, usada con un significado específico, tiende a co-ocurrir con las mismas palabras en su entorno. Luego, nuevas ocurrencias de la palabra a desambiguar se clasifican en los *clusters* inducidos.

Nótese que a diferencia de WSD, donde se busca la asignación explícita de una etiqueta de significado a una palabra objetivo, WSI realiza una discriminación de significados entendida como el agrupamiento de ocurrencias de una palabra en cierto número de clases de acuerdo a la determinación, por parte del algoritmo utilizado, de cuales ocurrencias corresponden a una misma clase (significado), únicamente a partir de estadísticas de co-ocurrencia entre palabras, sin que se tenga certeza de que las clases (significados) inducidas tengan un significado correspondiente en un inventario de significados determinado, como se muestra en la fig. 9.

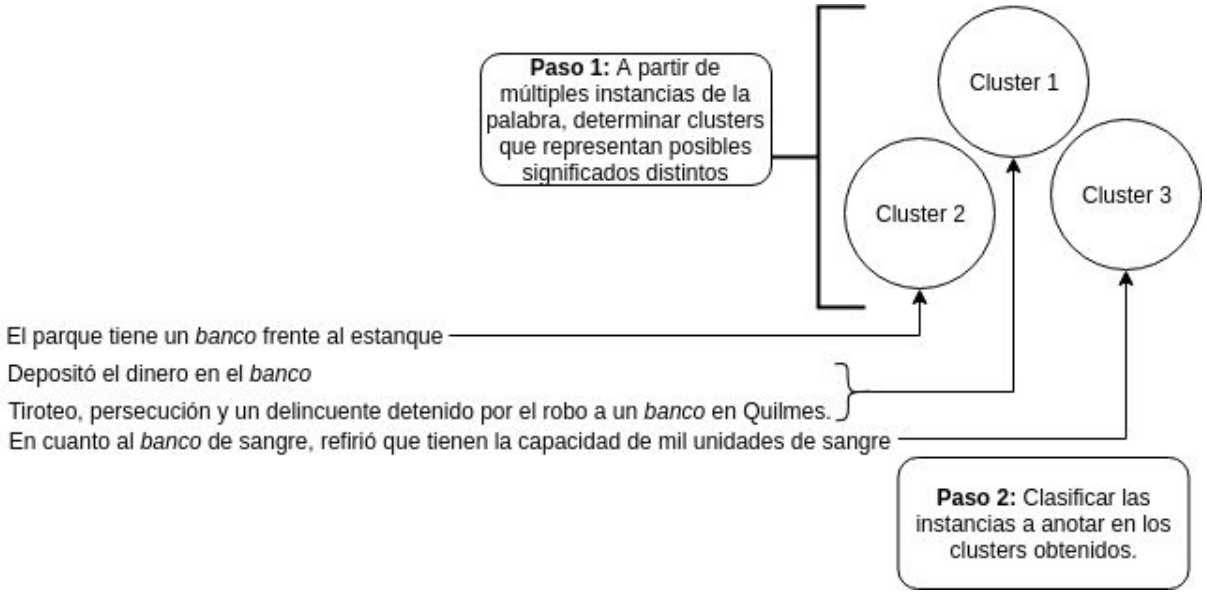

Fig. 9: Ejemplo de WSI para 4 instancias de la palabra *banco*. Ambos pasos pueden ser hechos a la vez, utilizando las propias instancias a anotar para obtener los clusters.

2.3.3.1. Principales enfoques

Los principales enfoques [Navigli, 2012] para la implementación de WSI son:

- Clusterización de contexto: en este enfoque cada ocurrencia de una palabra a desambiguar en un corpus se representa como un vector de contexto. Los vectores son luego clusterizados, donde cada cluster representa un significado de la palabra objetivo.
- Clusterización de palabra: consiste en clusterizar palabras semánticamente similares que expresan un significado de la palabra objetivo.
- Grafos de co-ocurrencia: estos métodos se relacionan con los anteriores, pero construyen grafos de co-ocurrencia de palabras para identificar el conjunto de significados de una palabra dada. Las co-ocurrencias entre palabras puede obtenerse en base a relaciones gramaticales o posicionales.
- Clusterización probabilística [Brody y Lapata, 2009]: utilizan un enfoque probabilístico, formalizando WSI en un modelo generativo. Para cada palabra ambigua, se modela el contexto como muestras de una distribución de probabilidad

sobre los sentidos, los que son a su vez modelados como una distribución de probabilidad sobre palabras.

- Latent semantic word spaces [Van de Cruys y Apidianaki, 2011]: la inducción y la desambiguación se basan en mapear las palabras y los contextos a un número limitado de dimensiones temáticas en un espacio semántico latente de palabras. Un significado particular está asociado con una temática, y así los diferentes significados se discriminan por su asociación a una dimensión temática en particular. De la misma forma, una instancia particular se desambigua determinando su dimensión temática más relevante.

2.3.3.2. WSI probabilístico

En particular, el método sobre el que se profundizará en este trabajo puede considerarse como una forma de clusterización ponderada de representantes de la palabra a desambiguar, obtenidos de considerar la desambiguación como una tarea de sustitución léxica, construyendo un sistema que reemplace a la palabra a desambiguar, manteniendo el significado de la misma tanto como sea posible.

Puede verse que, dado que su resultado es una clusterización probabilística de las instancias, el método es capaz de resolver un problema más complejo que el WSI típico, que podría considerarse como WSI ponderado.

A forma de adelanto, y como ejemplo para el lector, dado un conjunto de oraciones que contienen diversas formas del verbo *"apuntar"*, se busca para cada una de ellas conjuntos de posibles sustitutos de la palabra objetivo en la oración, de forma que para *"el estudiante apuntó en su cuaderno"* sería deseable contar con conjuntos donde aparezcan palabras como "*anotar*", *"escribir*", etc., mientras que para la oración "*apuntó al culpable con su dedo*" sería esperable obtener conjuntos de candidatos a sustitutos que contuviesen palabras como "*señalar*" o "*indicar*". Luego, clusterizando los conjuntos de acuerdo a su similitud en cuanto a la palabras en común, aquellas instancias de la palabra objetivo que refieren a un mismo significado tendrán sus conjuntos de representantes en una misma agrupación (*cluster*). La ponderación o probabilidad de la pertenencia de una instancia a un cluster, se deriva de la cantidad de conjuntos de sustitutos para la instancia que fueron asignados a dicho cluster.

2.3.3.3. WSI en español

Múltiples trabajos disponibles aplican técnicas de todos los tipos mencionados para realizar WSI en el idioma inglés. Sin embargo para el español, los esfuerzos se han concentrado en WSD. Durante la recopilación de trabajos previos en el área para la realización de este trabajo, no se pudo encontrar resultados disponibles sobre WSI para el español. En las evaluaciones internacionales SemEval (y sus antecesoras, Senseval), donde se evalúa el estado del arte en distintos problemas de procesamiento de lenguaje natural, no hay antecedentes de tareas de WSI para el español.

2.3.3.4. Técnicas de WSI

Primeramente, en esta sección se presenta una primera aproximación a la resolución del problema de WSI.
A continuación, se describen brevemente las técnicas AI-KU [Baskaya et al., 2013] y el modelo de Amrami y Goldberg [2018] que se utilizará en este trabajo (y está inspirado en el anterior).

Primera aproximación: clusterización de palabras
Una aproximación inicial a cómo determinar los posibles significados de una palabra sin contar con un inventario de significados dado, consiste en aplicar algoritmos de clusterización a las palabras representadas por vectores de *features*, usando una función de similaridad basada en la información mutua puntual.
Por ejemplo, puede considerarse la definición de información mutua utilizada por Pantel y Lin [2002] para un contexto *c* y un conteo de frecuencias $F_c(w)$ de la palabra *w* que ocurre en el contexto *c* :

$$mi_{w,c} = \frac{\frac{F_c(w)}{N}}{\frac{\sum_i F_i(w)}{N} \times \frac{\sum_J F_c(j)}{N}} \times \frac{F_c(w)}{F_c(w)+1} \times \frac{min(\sum_i F_i(w),\ \sum_j F_c(j))}{min(\sum_i F_i(w),\ \sum_j F_c(j))+1}$$

Donde $N = \sum_i \sum_j F_i(j)$ el total de conteos de frecuencia de todas las palabras y sus contextos, y una función de similaridad entre w_i y w_j, en este caso la similitud coseno:

$$sim(w_i, w_j) = \frac{\sum_c mi_{w_i c} \times mi_{w_j c}}{\sqrt{\sum_c mi^2_{w_i c} \times \sum_c mi^2_{w_j c}}}$$

Luego puede usarse cualquier algoritmo de clusterización a los vectores de *features* de las palabras, por ejemplo *K-means* donde a cada elemento se asigna a uno de K clusters de acuerdo a cual centroide se encuentra más cercano según la función de similaridad y cada centroide es recalculado como como el promedio de los elementos del cluster repitiendo el proceso hasta que el modelo converge.
Supongamos que queremos desambiguar 3 instancias de ***bomba***, como se ve en la fig. 10:

- ***Bomba.1****: El septiembre de ese mismo año, colocaron* ***bombas*** *en el hotel Hilton.*
- ***Bomba.2****: La viscosidad del anticongelante, y con ella la resistencia que éste opone a la* ***bomba****, depende de la temperatura de manera no lineal.*
- ***Bomba.3****: Menos mal en Cancún los zapatistas no pusieron* ***bombas.***

Fig. 10: Ejemplo para desambiguación de 3 instancias de "*bomba*".

En primera instancia necesitamos definir los vectores de *features* a utilizar, típicamente vectores de alta dimensión con baja densidad, para modelar la aparición o no de palabras en una ventana determinada tomada a partir de la palabra objetivo, *POS-tags* de las

palabras próximas, etc. Con los conteos de frecuencia de dichos vectores se puede calcular la información mutua entre palabra y contexto, y con esta la similaridad entre palabras (así como con el centroide de los clusters).
La primera etapa del método consiste en tomar una gran cantidad de instancias de la palabra ***bomba***, y obtener, mediante *K-means* una clusterización de las mismas utilizando la función de similitud. El resultado será k clusters representando cada uno un significado posible de ***bomba***. Luego, para cada uno de las instancias ***bomba1***, ***bomba.2*** y ***bomba.3***, utilizando la función de similitud, se asigna la instancia al cluster cuyo centroide minimiza la distancia coseno.

AI-KU: Vectores de sustitutos y modelado de coocurrencia para WSI
Este trabajo, propuesto por Baskaya et al. [2013], a diferencia de los métodos basados en la clusterización o partición de grafos sobre una representación de las coocurrencias de una palabra, propone crear vectores de sustitutos para cada palabra objetivo a partir de los sustitutos más probables sugeridos por un modelo de lenguaje estadístico. Las muestras de sustitutos para formar estos vectores se toman de acuerdo a las probabilidades de aquellos y los resultados del modelo de co ocurrencia son luego clusterizados.
La representación del contexto con las coocurrencias de una palabra puede ser dificultosa dada la gran cantidad de factores a considerar, como el orden de ocurrencia, el tamaño de la ventana de contexto, etc. En lugar de lidiar con todos estos factores, sugieren representar el contexto mediante los sustitutos más probables que determine el modelo de lenguaje (idea que se mantiene en el método de *Symmetric Patterns* de Amrami y Goldberg que se utilizará en este trabajo). Estos sustitutos se usan para crear pares de *(identificador de instancia, palabra sustituta)* para alimentar el modelo de coocurrencia. La salida de los modelos de co ocurrencia es luego clusterizada utilizando el algoritmo de *k-means*. Estos pasos se aplican para cada palabra objetivo separadamente.
Los sustitutos son obtenidos de un modelo de lenguaje de 4-gramas construido utilizando el corpus *ukWaC*, adicionado a las instancias de Semeval-2013 *Task* 13. Para cada instancia en este corpus expandido, se toman 3 palabras a la izquierda y 3 a la derecha de cada palabra objetivo, como contexto para estimar las probabilidades de los potenciales sustitutos léxicos. Un tamaño de ventana acotado permite obtener mejores resultados en cuanto a la similaridad semántica, mientras que los contextos extensos son mejores para modelar relaciones temáticas.
Se consideran los 100 sustitutos con mayor probabilidad, normalizando su probabilidad, obteniéndose un vector de sustitutos para cada instancia a desambiguar. El vector de sustitutos es entonces una función únicamente del contexto y es independiente de la palabra objetivo, a diferencia de lo que sucede en *Symmetric Patterns*. Luego, para cada instancia se muestrean 100 sustitutos de su vector de sustitutos.
Los sustitutos obtenidos, se utilizan para formar los pares *(identificador instancia, sustituto)*. Si dos instancias tienen pares de palabras sustitutas en común, esto implica que tienen contextos similares, y esos pares de palabras se “atraen” al momento de la clusterización, y en caso contrario se “repelerán”. El proceso de muestreo y clusterización se realiza para cada palabra objetivo.

Luego de este proceso, se cuentan las etiquetas de clase para cada instancia. Si para una instancia, 50 pares fueron asignados a la clase c_1, 30 a la clase c_2 y 20 a la clase c_3, se interpreta como que el sentido 1 tiene una aplicabilidad de 50%, el 2 de un 30% y el 3 de un 20%.

Supongamos que queremos desambiguar las 3 instancias de ***bomba*** que se presentaron en la fig. 10. Se consulta al modelo de lenguaje de 4-gramas con las entradas:

- **1.1.** mismo año colocaron __
- **1.2.** __ en el hotel
- **2.1.** opone a la __
- **2.2.** __ depende de la
- **3.1.** zapatistas no pusieron __

Digamos que se consideran los 4 sustitutos de mayor probabilidad (a diferencia de los 100 utilizados en el método), entonces tenemos un conjunto de sustitutos con su probabilidad normalizada para cada instancia:

- **1.** *{artefactos: 0.5, explosivos: 0.3, dinamita: 0.1, atentado: 0.1}*
- **2.** *{presión: 0.35, fuerza: 0.25, volumen: 0.2, potencia: 0.2}*
- **3.** *{proyectiles: 0.3, bengalas: 0.3, armas: 0.2, ametralladoras:0.2}*

Y de estos sustitutos, se muestrean 3 por instancia de acuerdo a su probabilidad (el método muestrea 100 en su versión completa)

- **1.** *artefactos, explosivos, dinamita*
- **2.** *presión, volumen, potencia*
- **3.** *bengalas, armas, proyectiles*

Con esto tenemos los pares de identificador de la instancia y sustituto:

(1, artefactos), (1, explosivos), (1, dinamita), (2, presión), (2, volumen), (2, potencia), (3, bengalas), (3, armas), (3, proyectiles)

Para las palabras muestreadas se obtienen sus embeddings S-CODE [Maron et al., 2010] y estos se clusterizan utilizando *k-means*. Supongamos que tenemos los 3 representantes de la instancia 1 en el mismo cluster que 2 representantes de la instancia 3. Llamemos a ese cluster 0, entonces tenemos que la instancia 1 pertenece al cluster 0 con ponderación 1, mientras que la 3 pertenece al cluster 0 con ponderación 0.66. Digamos que el restante representante de 3 pertenece a un cluster 1, entonces la instancia 3 pertenece al cluster 1 con ponderación 0.33. Por último supongamos que todos los representantes de la instancia 2 pertenecen a un cluster 3 y entonces dicha instancia pertenece a dicho cluster con ponderación 1.

Word Sense Induction con ELMO biLM y Symmetric Patterns

Continuando con la idea de emplear sustitutos de las palabras objetivo generados por un modelo de lenguaje, para luego inducir los significados a partir de su clusterización, Amrami y Goldberg [2018] utilizan en este trabajo un modelo de lenguaje basado en una red neuronal recurrente (RNN) en lugar de los tradicionales modelos de lenguaje basados en n-gramas. El modelo de lenguaje basado en RNN permite además una segunda variación realizada por los autores, que estos denominan *Symmetric Patterns*, que se expondrá junto con una descripción detallada del algoritmo en la sección 3.1.

2.3.4. SemEval-2013 Task 13

La tarea 13 de SemEval-2013 evalúa sistemas de WSD y WSI en un contexto donde las ocurrencias pueden ser etiquetadas con múltiples significados, ponderados de acuerdo a su aplicabilidad [Jurgens y Klapaftis, 2013].
Se centra en la desambiguación de significados para 50 lemas objetivo a desambiguar (20 sustantivos, 20 verbos y 10 adjetivos). Los sistemas participantes se evalúan en dos configuraciones dependiendo de si utilizan significados inducidos (WSI) o significados tomados de WordNet[12] (WSD) para realizar la clasificación.
La tarea requiere que los sistemas participantes anoten las instancias de los lemas objetivo usando WordNet debido a la alta granularidad de sus significados, pudiendo anotar una instancia con uno o más significados ponderados por su probabilidad.
Los sistemas de WSI participantes fueron entrenados con el corpus ukWaC[13] para la inducción de significados. A diferencia de tareas previas de WSI en SemEval, donde se proveía a los participantes de corpus específicos para los términos de la tarea, en este caso se utiliza un corpus de gran extensión para favorecer a los métodos de WSI con una mayor cantidad de datos de los que tomar asociaciones estadísticas.
Los autores de la tarea plantean realizar la evaluación en dos partes. Para los sistemas de WSI, las etiquetas sobre los significados inducidos se convierten a etiquetas de WordNet mediante un procedimiento de mapeo. En una segunda evaluación, se realiza una comparación directa de los dos inventarios de significados mediante comparaciones de clusterización.

2.3.4.1. Evaluación en WSD

En la primera evaluación, se considera una tarea de WSD con tres objetivos, cada uno con una medida específica:

1. Detectar los significados aplicables. Para su medición se utiliza el Índice Jaccard: dados dos conjuntos de etiquetas de significado para una instancia, X e Y, mide el acuerdo como $|X \cap Y| \ / \ |X \cup Y|$. El valor alcanza el máximo cuando X e Y contienen etiquetas idénticas, y es mínimo cuando ambos conjuntos son disjuntos. Por ejemplo, supongamos que tenemos una instancia etiquetada por el sistema con las etiquetas A, B y C, independientemente del orden y la ponderación que le asigne a cada una, y que para dicha instancia el *gold standard* indica que las etiquetas correctas, nuevamente independiente del orden y la ponderación, son A, C y D. Entonces tenemos que la cardinalidad de la intersección es 2 y la cardinalidad de la unión de ambos conjuntos de etiquetas es 4, por lo que el Jaccard Index para el sistema en esa instancia es 0.5.
2. Ponderarlos de acuerdo a su aplicabilidad. Como métrica se utiliza la Similaridad Tau de Kendall ponderada por posición: la distancia Tau de Kendall, K, es una medida de la cantidad de cambios de posición de ítems necesarios para que dos

[12] Versión 3.1
[13] https://www.sketchengine.eu/ukwac-british-english-corpus/

secuencias sean idénticas. La definición se extiende usando una función de penalidad *δ* para el costo de intercambiar dos posiciones, denotada K_{δ},. Usando una δ apropiada, K_{δ} se puede sesgar hacia la correctitud de las posiciones más altas, asignando valores de δ más pequeños a las posiciones más bajas. K_{δ} es una medida de distancia, por lo que su rango de valores depende del número de posiciones. La distancia se normaliza entonces a [0,1] dividiendo por el máximo de K_{δ} y restando el resultado a 1. Dadas dos secuencias *x* e *y*, donde *x* es la referencia para medir *y*, la similaridad normalizada es: $K_{\delta}^{sim} = K_{\delta}(x,y) / K_{\delta}^{max}(x)$. Los autores definen *δ* de forma que el costo de mover un elemento a una posición más alta sea mayor, usando la fórmula $\delta_i=(n-(i+1))/n$, donde *n* es el número de sentidos e *i* la posición a la que se mueve un item al reordenar la secuencia.

Por ejemplo, en el caso donde tenemos una secuencia de etiquetas producida por el sistema A, B, C, para evaluar en comparación a una secuencia de etiquetas en el *gold standard* A, C, D, tenemos un vector *δ = [1, 0.75, 0.5, 0.25],* a partir del cual tenemos un vector de penalidades para permutaciones en cada distancia posicional donde cada posición es 1 más la suma de las anteriores en *δ, p = [1, 2, 2.75, 3.25]*. Agreguemos a cada secuencia, la etiqueta faltante al final, de forma que la secuencia del sistema resulta, A, B, C, D y la del *gold standard* A, C, D, B. Comenzando por el primer elemento del *gold standard*, A, se busca su posición en la secuencia producida por el sistema, como también se encuentra en la primera posición, en este caso la distancia es 0. Pasando al segundo elemento del *gold standard* se encuentra C, mientras que en la secuencia del sistema la segunda posición la ocupa B, que es la cuarta posición del *gold standard*, por lo que el costo para este intercambio es $\delta_2 \times (p_4 - p_2)/(4 - 2)$ o sea, 0.75 x (3.25 - 2) / (4 - 2) = 0.47. El tercer elemento de la secuencia del *gold standard* es D, que es la cuarta posición en la secuencia del sistema, posición que en el *gold standard* es ocupada por B, segunda posición en la secuencia del sistema, por lo que el costo de este intercambio es 0.31, lo que sumado al 0.47 anterior resulta en 0.78. Luego se calcula la máxima distancia, como el costo de intercambiar todos los pares *(i, j)* donde *i < j*, que en este caso es 3.375. Por último, decimos que el valor de *positional Kendall's* Tau normalizada es 1 - (0.78 / 3.375) = 0.77.

3. Medir el acuerdo de las ponderaciones con las anotadas por humanos. Como medida se utiliza una variante de *Normalized Discounted Cumulative Gain*: dada una distribución *gold-standard* de la aplicabilidad de los *k* sentidos, donde w_i es la probabilidad del sentido *i* en dicha distribución, y un secuencia ordenada a evaluar de los *k* sentidos donde $\hat{w}_i$ es la probabilidad del sentido *i*, se define *Discounted Cumulative Gain* como:

$$DCG = \sum_{i=1}^{k} \frac{2^{w_i+1}-1}{log_2(i+1)}$$

Para considerar los valores de aplicabilidad en la distribución a evaluar, se define *Weighted Discounted Cumulative Gain*:

$$WDCG = \sum_{i=1}^{k} \frac{\frac{min(w_i,\hat{w}_i)}{max(w_i,\hat{w}_i)}(2^{w_i+1}-1)}{log_2(i)}$$

Que luego es normalizado a [0,1] al dividir entre el máximo valor alcanzable de DCG, que se da al ordenar los *k* items por su ponderación de acuerdo al *gold standard*.
Para ejemplificar, consideremos nuevamente el caso en que la salida del sistema correspondía a la secuencia A, B, C, pero teniendo en cuenta que para esta métrica es relevante la ponderación de cada etiqueta, supongamos que las ponderaciones son {A: 0.6, B: 0.3, C: 0.1}, y análogamente para el *gold standard* supongamos que tenemos {A: 0.7, C: 0.2, D: 0.1}. Para cada posición de la secuencia del sistema, conteniendo la etiqueta X, se calcula el valor de:

$$\frac{min(ponderación_{gold}(X), ponderación_{sistema}(X))}{max(ponderación_{gold}(X), ponderación_{sistema}(X))} \times (2^{1+ponderación_{gold}(X)} - 1)$$

En el caso de nuestro ejemplo: 1.93, 0 y 0.65. Luego estos valores se suman multiplicados por $1 / log_2(i+1)$ donde *i* es la posición en la secuencia del sistema para cada uno, para obtener el *Discounted Cumulative Gain* (DCG), que en este caso vale 2.255. Por otro lado para la secuencia del *gold standard* se calcula

$$\sum_i \frac{2^{1+ponderación_{gold}(X_i)} - 1}{log_2(i+1)}$$

Cuyo valor en nuestro ejemplo es 5.77, y dividiendo DCG sobre este valor se tiene el *Weighted Normalized Discounted Cumulative Gain*, que en el caso del ejemplo es 0.39.

Para la evaluación en WSD, las etiquetas de significados inducidos deben ser convertidas a un inventario de referencia (WordNet) mediante un procedimiento de mapeo. Para esto se sigue el procedimiento sugerido por Jurgens [2012], el cual se basa en Agirre [2006], utilizando el mecanismo de partición del corpus implementado por Manandhar [2010] en 5 particiones, 4 de las cuales se usan para aprender el mapeo entre etiquetas inducidas y las utilizadas en la *gold key* de la tarea. La restante partición es luego convertida utilizando ese mapeo y comparado con el *gold standard*. El proceso se repite, rotando la partición a mapear, de forma que cada partición sea evaluada una vez.

2.3.4.2. Evaluación en WSI

Para la segunda parte de la evaluación, donde se busca evaluar la "calidad" de la clusterización de las instancias para el problema de WSI, se proponen dos nuevas medidas para evaluar asignaciones de significados en un contexto donde los conjuntos de instancias se superponen y las instancias pueden pertenecer a dichos contextos parcialmente (de acuerdo a su aplicabilidad, la probabilidad de que la instancia pertenezca al conjunto), lo que es llamado por los autores como "*fuzzy cover*".
Las medidas propuestas son *Fuzzy B-Cubed*, que sumariza el rendimiento por instancia proveyendo un estimado de qué tan bien rendiría el sistema en un nuevo corpus con una distribución de significados similar, y *Fuzzy NMI*, medida sobre los clusters obtenidos en lugar de sobre las instancias, y por lo tanto dando un indicador del rendimiento del método de forma independiente a la distribución de significados en el corpus[14].

[14] Una descripción detallada del cálculo de estas métricas, junto con ejemplos, puede encontrarse en el Anexo B.

B-Cubed es una medida basada en precisión y recall, que estima el ajuste entre dos clusterings *X* e *Y* a nivel de items. *Fuzzy B-Cubed* es una generalización de esta medida para *fuzzy covers*.
La métrica *Fuzzy Normalized Mutual Information (FNMI)*, deriva de la información mutua definida formalmente como $I(X,Y)=H(X)-H(X|Y)$ donde $H(X)$ es la entropía de la variable aleatoria *X* que representa los conjuntos de instancias asignados a cada sentido. La información mutua típicamente se normaliza al intervalo [0,1] de forma de facilitar la comparación entre distintas clusterizaciones en una escala común, utilizando $Max(H(X),H(Y))$ como factor normalizador.
.

3. Implementación

En esta sección se detallan los pasos seguidos en la implementación llevada a cabo para probar el método de *WSI with Neural biLM and Symmetric Patterns* en español, más específicamente sobre la ya presentada tarea de *Spanish Lexical Sample* de *Senseval 2*.
En la primera subsección se desarrolla en detalle el método de Amrami y Goldberg [2018]. Las siguientes subsecciones detallan la implementación realizada en este trabajo.
La implementación puede dividirse en 2 partes, el entrenamiento del modelo de lenguaje biLM ELMo incluyendo el procesamiento del corpus necesario para entrenar y los cambios realizados al modelo, y por otro, los cambios realizados al algoritmo de desambiguación con *Symmetric Patterns*.

3.1. Word Sense Induction con ELMO biLM y Symmetric Patterns

El método se basa en vectores sustitutos que representan cada instancia de la palabra a desambiguar como una distribución de posibles palabras sustitutas, determinadas por el modelo de lenguaje, los cuales luego se clusterizan para obtener los significados.
Los autores tomaron el modelo de lenguaje basado en una red neuronal ELMo biLM de Peters et al. [2018], el cual ha sido probado con buenos resultados en varias tareas de procesamiento de lenguaje natural. En lugar de utilizar los vectores de estado de la LSTM como sugieren los autores de ELMo, se utilizan las probabilidades de las palabras sustitutas obtenidas del modelo de lenguaje.
Dada una palabra objetivo, lematizada y con su etiqueta de categoría gramatical (POS-*tagging*), con un conjunto de oraciones en los cuales la palabra es utilizada (instancias), el objetivo es clusterizar las distintas ocurrencias de la palabra objetivo de forma que cada cluster corresponda a un significado diferente de la palabra. De acuerdo a lo establecido por la tarea 13 de SemEval 2013, se busca una clusterización ponderada, en la que cada ocurrencia de la palabra sea etiquetada con una probabilidad de pertenencia a cada uno de los clusters (significados).
Al experimentar con este modelo, los autores notaron que si bien los resultados obtenidos mejoraban los del sistema AI-KU de Baskaya et al. para la tarea 13 de SemEval 2013, los sustitutos no tomaban en cuenta la palabra a desambiguar en sí misma, dado que al consultar el modelo de lenguaje este solo recibe las palabras previas o posteriores a la palabra a objetivo (según se trate del modelo *forward* o *backward,* respectivamente). Para proveer al modelo de información semántica sobre la palabra a desambiguar incorporaron una técnica que denominan *Symmetric Patterns*.

Symmetric Patterns
Esta técnica se basa en aprovechar las capacidades del modelo bidireccional de lenguaje neuronal de forma de obtener sustitutos dependientes del contexto que sigan un patrón

simétrico en términos semánticos. Dada las aptitudes para la generalización de los modelos de lenguaje neuronales y la abundancia del patrón "X y Y" [15]("*X and Y*" en inglés) en el lenguaje natural, se consulta al modelo de lenguaje con un patrón incompleto creado dinámicamente y se le pide predecir los candidatos probables para completarlo.
Considerando la oración de ejemplo: "*Me gustó el sonido del clavicordio*" donde la palabra a desambiguar es *"sonido"*, en un método basado únicamente en el contexto (como el propuesto en el sistema AI-KU), se tendrían las distribuciones[16]:

LM→(<s>Me gustó el _)
LM←(_del clavicordio</s>)

Al condicionar al modelo de lenguaje únicamente por el contexto se omite información valiosa. En el caso del ejemplo, no se provee ninguna información al modelo de lenguaje sobre qué característica del clavicordio hace referencia la oración, podría tratarse de su color, su tamaño, etc. lo que claramente daría sustitutos muy distintos a los que se buscan, que deberían vincularse con su sonido. Para esto se modifica la entrada al modelo de forma de obtener de este las distribuciones correspondientes a $p\rightarrow(w'|w_1,...,w_i,"y")$ y $p\leftarrow(w'|w_n,...,w_i,"y")$ (nótese que el símbolo "*y"* no se está usando como una variable si no para denotar la conjunción copulativa "y" del idioma español). En el caso de ejemplo se tiene:

LM→(<s>Me gustó el sonido y _)
LM←(_ y sonido del clavicordio</s>)

Los autores afirman que de esta forma se obtienen distribuciones más acertadas que incorporan tanto el contexto global como información sintáctica local provista por la palabra objetivo.

Ejemplo de aplicación del algoritmo

Supongamos que queremos desambiguar 3 instancias de la palabra "***bomba***":

- ***Bomba.1****: El septiembre de ese mismo año, colocaron* ***bombas*** *en el hotel Hilton.*
- ***Bomba.2****: La viscosidad del anticongelante, y con ella la resistencia que éste opone a la* ***bomba****, depende de la temperatura de manera no lineal.*
- ***Bomba.3****: Menos mal en Cancún los zapatistas no pusieron* ***bombas.***

[15] Si bien en español la conjunción "*y*" es reemplazada en determinadas ocasiones por "*e*", estos casos no presentan una diferencia a la hora de considerar la oración con sus palabras lematizadas (el lemma de "*e*" es "*y*"), y aún en el caso en que se consideren las oraciones sin lematizar, el consultar al modelo de lenguaje con "*e*" podría condicionar los sustitutos a predecir por el modelo de lenguaje.
[16] Los símbolos especiales *<s>* y *</s>* son utilizados en el modelo biLM ELMo para indicar el comienzo y el fin de la oración respectivamente.

Obtención de sustitutos mediante biLM ELMo

Tomemos la instancia ***bomba.1***. Tal como se dijo anteriormente, mediante la aplicación de *symmetric patterns*, se presentará al modelo de lenguaje *forward* la oración:

El septiembre de ese mismo año, colocaron bombas y ___

Dado que el método de desambiguación por defecto aplica lematización, la oración recibida por el modelo forward será en realidad:

El septiembre de ese mismo año, colocar bomba y ___

Donde el modelo de lenguaje deberá responder con los sustitutos más probables para agregar como última palabra de la oración. Análogamente el modelo de lenguaje *backward* recibirá la oración:

___ y bombas en el hotel Hilton.

En su versión lematizada:

___ y bomba en el hotel Hilton

Y deberá arrojar los sustitutos de mayor probabilidad para colocar como primera palabra de la oración.

Supongamos entonces que el modelo *forward* sugiere los siguientes sustitutos, con sus probabilidades:

[*'explosivo': 0.49, 'misil': 0.18, 'ametralladora': 0.21, 'cohete': 0.12*]

Y el backward:

[*'bengala': 0.05, 'explosivo': 0.31, 'proyectil': 0.23, 'artillería': 0.23, 'mortero': 0.18*]

Formación de representantes

El siguiente paso, consiste en formar conjuntos, llamados representantes, a partir de los sustitutos sugeridos por los modelos de lenguaje. Esto se hace seleccionando aleatoriamente de los conjuntos de sustitutos ponderados por las probabilidades. Por defecto, en el método se forman 20 representantes por instancia, formados por 4 palabras muestreadas del modelo *forward* y 4 del *backward*. Para los fines de esta explicación, consideremos que se utilizan 3 representantes por instancia, formados por 1 palabra del

modelo *forward* y 1 del *backward*. Supongamos entonces, que luego de realizado este procedimiento para ***bomba.1*** tenemos los representantes:

Bomba.1.1: *{'explosivo': 2 }*
Bomba.1.2: *{'misil': 1, 'proyectil':1}*
Bomba.1.3: *{'explosivo': 1, 'proyectil':1}*

Nótese que se indica la multiplicidad con la que la palabra aparece en el representante, ya que una misma palabra puede resultar "sorteada" para un mismo representante en más de una ocasión. Análogamente, para las instancias ***bomba.2*** y ***bomba.3*** tendremos también 3 representantes de 2 palabras para cada uno de ellos, obtenidos por idéntico procedimiento a partir de los candidatos a sustitutos arrojados por los modelos de lenguaje para cada instancia:

Bomba.2.1: *{'tanque': 1, 'generador': 1}*
Bomba.2.2: *{'cilindro': 1, 'tanque': 1}*
Bomba.2.3: *{'generador': 1, 'tanque': 1}*

Bomba.3.1: *{'explosivo': 1, 'tanque': 1}*
Bomba.3.2: *{'explosivo': 2}*
Bomba.3.3: *{'tanque': 1, 'ojiva': 1}*

Clusterización de representantes

En base a estos representantes, se construye una matriz con una fila para cada representante y una columna para cada palabra en los representantes, donde cada celda contiene el número de apariciones de la palabra en el representante:

	explosivo	**misil**	**proyectil**	**ojiva**	**tanque**	**generador**	**cilindro**
bomba.1.1	2	0	0	0	0	0	0
bomba.1.2	0	1	1	0	0	0	0
bomba.1.3	1	0	1	0	0	0	0
bomba.2.1	0	0	0	0	1	1	0
bomba.2.2	0	0	0	0	1	0	1
bomba.2.3	0	0	0	0	1	1	0
bomba.3.1	1	0	0	0	1	0	0
bomba.3.2	2	0	0	0	0	0	0
bomba.3.3	0	0	0	1	1	0	0

Luego sobre esta matriz se aplica el proceso de clusterización de las filas (transformación a frecuencias relativas y clusterización aglomerativa con distancia coseno y *average linkage*). En este caso, el resultado sería la siguiente asignación de clusters[17]:

	Clúster
bomba.1.1	1
bomba.1.2	1
bomba.1.3	1
bomba.2.1	0
bomba.2.2	0
bomba.2.3	0
bomba.3.1	1
bomba.3.2	1
bomba.3.3	0

De esta forma tenemos que los 3 representantes de ***bomba.1*** pertenecen al cluster 1, por lo que decimos que la ponderación de la pertenencia de ***bomba.1*** al cluster 1 es 1. Para ***bomba.2*** sus 3 representantes pertenecen al clúster 0, por lo que la ponderación de su pertenencia al clúster 0 es 1. Para ***bomba.3*** tenemos que 2 representantes (***bomba.3.1*** y ***bomba.3.2***) pertenecen al clúster 1, mientras que 1 (***bomba.3***) pertenece al clúster 0, por lo tanto decimos que pertenece al clúster 0 con ponderación 0.33 y al 1 con ponderación 0.66.
De esta forma, puede inferirse que ***bomba.1*** y ***bomba.3*** comparten un mismo significado de la palabra "*bomba"*, como cabía esperar, mientras que ***bomba.2*** corresponde a otra acepción de la palabra. Nótese que el desambiguador obtiene probabilidades de pertenencia al clúster, que permite denotar casos donde a una instancia le apliquen distintos significados de la palabra en distinto grado, es decir, donde es más ambigua su utilización.

[17] La clusterización fue realizada con la librería SciPy de Python

3.2. Entrenamiento

El código de la red neuronal para el entrenamiento de biLM ELMo utilizando la librería *TensorFlow* creado por Peters et al. [2018] se encuentra disponible para su descarga[18].
Para llevar a cabo el entrenamiento del modelo de lenguaje con los recursos disponibles se realizaron modificaciones que se detallan a continuación. Más adelante se detallan los resultados obtenidos y otros datos derivados del entrenamiento.

3.2.1. Corpus

Para el entrenamiento del modelo biLM ELMo se partió del *Spanish Billion Word Corpus and Embeddings*, una colección de textos sin anotar junto con embeddings obtenidos a partir del mismo [Cardellino, 2016].
El corpus contiene cerca de 1.500 millones de palabras, recopiladas a su vez de diferentes corpus y recursos en internet. A su vez, se encuentra disponible junto con el corpus el conjunto de embeddings obtenido de este utilizando el algoritmo *word2vec*. Para los efectos del presente trabajo solo fue utilizado el corpus sin anotar, para luego de un escaso preprocesamiento entrenar la red neuronal de biLM ELMO.
El corpus disponible para descargar contiene todos los caracteres no alfanuméricos reemplazados por espacios y los números reemplazados por el token “DIGITO”, así como las ocurrencias de más de un espacio de forma consecutiva por un único espacio.

3.2.1.1. Pre procesamiento de Spanish Billion Word Corpus

Dado que el tamaño del *Spanish Billion Word Corpus*, tanto por su extensión total como por la de su vocabulario, resultaba en un entrenamiento del modelo biLM ELMo demasiado extenso en el tiempo y demandante de una cantidad considerable de recursos computacionales, se decidió realizar un cierto pre procesamiento del mismo para volver el entrenamiento más rápido y menos costoso en recursos, depurando al mismo tiempo el corpus al eliminar algunas partes del mismo que pudieran aportar ruido al modelo.
Para utilizarlo como corpus de entrenamiento y evaluación del modelo de lenguaje biLM ELMo, se pre procesó el corpus completo siguiendo los siguientes pasos:

1. Eliminación de oraciones que contenían palabras en otro alfabeto que no fuera el latino. De esta forma podía reducirse el vocabulario sobre el que entrenaría el modelo, eliminando palabras que aportarían ruido y no serían luego sustitutos adecuados en el procedimiento de WSI. Dado que la eliminación de al menos una palabra de la oración la afectaría en su totalidad (sintáctica y semánticamente), y que el corpus era suficientemente extenso como para eliminar dichas oraciones sin reducir la utilidad del mismo (para los fines del entrenamiento), se decidió eliminar las oraciones completas. Para este fin se implementó el script *foreignTokenCleaner.py*. En total, se eliminaron 959.822 ocurrencias de palabras con caracteres no latinos.

[18] https://github.com/allenai/bilm-tf

2. Se experimentó con distintos tamaños de vocabulario, lo que a su vez condiciona el tamaño del corpus disponible, buscando maximizar el tamaño del mismo sin que el entrenamiento requiriera mayores recursos que los disponibles o un tiempo de ejecución demasiado extenso para los fines de este trabajo. El vocabulario completo del corpus, junto con la cantidad de apariciones de cada palabra, se obtuvo mediante la ejecución del script *corpus2freq.py*. Finalmente, como resultado de las pruebas realizadas con varios tamaños de vocabulario, se optó por un tamaño de vocabulario en el orden de 2^{17} (131.072), para el cual se consideró que se alcanzaba un equilibrio adecuado entre el tamaño del corpus disponible y del tiempo y recursos necesarios para llevar a cabo el entrenamiento. Para ello se ejecutó el script *corpusCleaner.py* con el vocabulario completo del corpus truncado de forma que sólo las 131.069 palabras con más apariciones en el corpus fueran consideradas (biLM ELMo utiliza 3 símbolos reservados que deben incluirse en el vocabulario). Cualquier oración que contuviera palabras fuera de ese vocabulario era descartada entonces por el programa, que a su vez realiza un reordenamiento aleatorio de las oraciones que serán seleccionadas para el corpus definitivo (esto es una recomendación de los autores de biLM ELMo). La salida de este script es: un corpus de entrenamiento conteniendo el 80% de las oraciones seleccionadas, un *held out* corpus para evaluación del modelo de lenguaje conteniendo el 20% restante de las oraciones seleccionadas, un archivo de vocabulario en el formato requerido para el entrenamiento de biLM ELMo junto con otro archivo de vocabulario conteniendo la cantidad de apariciones de cada palabra en el corpus.

Luego de este procesamiento se contaba con el corpus de entrenamiento conteniendo 614.901.945 palabras en 24.541.644 oraciones, y un corpus de evaluación con 153.725.405 palabras en 6.137.474 oraciones. El vocabulario final de este corpus resultante consta de 130.878 palabras diferentes (más los símbolos reservados: *</S>*, *<S>* y *<UNK>*[19]).

3.2.2. Parametrización del modelo de lenguaje

El código disponibilizado por los autores de biLM ELMo utiliza la librería Tensorflow para la implementación del modelo. Dicha librería cuenta con una versión optimizada para ejecutar en la GPU, reduciendo significativamente los tiempos de entrenamiento. Sin embargo, dados los recursos limitados en cuanto a memoria de la GPU, se realizaron las siguientes modificaciones a los parámetros del entrenamiento en el archivo *train_elmo.py*:

- El tamaño del batch (parámetro *batch_size*) se redujo a 64.
- El número de GPUs (parámetro *n_gpus*) se redujo a 1, dado que solo se contaba con una GPU (en el entrenamiento del modelo presentado por los autores en el trabajo publicado, el entrenamiento fue realizado en 3 GPUS NVIDIA GeForce GTX 1080).

[19] El símbolo *<UNK>* no se encuentra en el corpus, pero se agregó al vocabulario tal como especifican Peters et al. [2018]

- El número de tokens del corpus de entrenamiento fue ajustado de acuerdo a los contenidos en la partición destinada al entrenamiento del corpus procesado (parámetro *n_train_tokens*).
- El número de pasos de *unrolling* se redujo a 10 (parámetro *unroll_steps*).

Por otra parte, se modificó la función *dump_weights* de *training.py* que escribe un archivo en formato HDF5 con todos los pesos de la red resultantes del entrenamiento, ya que los autores evitaban expresamente que se incluyeran los pesos de la capa *softmax* dado que los mismos implican un aumento considerable del tamaño del archivo y son innecesarios para la obtención de embeddings, pero necesarios para la obtención de sustitutos (dado que se utilizará al modelo como modelo de lenguaje).

3.2.3. Ambiente de entrenamiento

El entrenamiento del modelo de lenguaje se realizó en un ambiente con las siguientes características:

Hardware:

- CPU Intel(R) Core(TM) i7-7700HQ CPU @ 2.80GHz
- 12GiB RAM
- 1 GPU NVIDIA GeForce® GTX 1050, 2 GB DDR5

Software:

- SO Ubuntu 16.04.5 LTS
- CUDA 9.0.176
- CuDNN 7.0.5
- Ambiente virtual Conda 4.5.11
    - Python 3.6.6
    - Tensorflow-gpu 1.5.0
    - NumPy 1.15.1

3.2.4. Resultados del entrenamiento

El entrenamiento del modelo insumió unos 94 días, finalizando con una perplejidad del modelo sobre el corpus de entrenamiento de 17,7893.

La perplejidad del modelo de lenguaje en el *held out* corpus fue de 28,9978. A modo de comparación, la perplejidad del modelo de lenguaje entrenado por los autores de ELMo es de 39,7. Si se compara con respecto al tamaño del vocabulario, el vocabulario del modelo entrenado para español contiene 131.072 palabras distintas, mientras que el utilizado por los autores consiste de 793.471, por lo que las relaciones perplejidad/tamaño de vocabulario son 0,00022 y 0,00005 respectivamente.

Como resultado del entrenamiento, se obtiene el archivo de *dump* de los pesos de la red neuronal, incluyendo los de la capa *softmax* que distribuye la probabilidad entre el vocabulario, por lo que puede utilizarse tanto para obtener embeddings como en forma de modelo de lenguaje.

Es importante en este punto destacar el hecho de que los pesos de la red entrenada se encuentran disponibilizados en forma pública para que puedan ser utilizados en trabajos

futuros sin necesidad de repetir el proceso de entrenamiento, por lo que son en sí mismos un resultado de valor derivado del presente trabajo.

Pruebas de similaridad con embeddings

A partir de los pesos de las capas de la red neuronal entrenada, es posible calcular embeddings, contextualizados o no. Como evaluación de la calidad de los embeddings resultantes se aplicaron pruebas de similaridad y analogía de palabras, para las cuales se obtenía su embedding sin contexto y luego se calculaba el coseno de los mismos como medida de similaridad. En el caso de las pruebas de similaridad, la correlación de Spearman entre la secuencia de cosenos de los pares de palabras y la secuencia de puntajes de similaridad asignado por la evaluación debería ser cercano a 1, mientras que en las pruebas de analogía se busca la palabra cuyo embedding sea el de menor coseno con el vector resultante de una operación matemática entre los embeddings de otras 3 palabras buscando encontrar patrones del tipo: *embedding(reina) - embedding(mujer) + embedding(hombre) ≅ embedding(rey)*.

En la Fig. 11 se reporta el valor de la correlación de Spearman entre el coseno de los *embeddings* de los pares de palabras en la evaluación y el *score* del *test* para el modelo entrenado comparado con el modelo *ELMo for Many Langs* de Che et al. [2018]

	ELMo Entrenado	ELMO for Many Langs
WS353v2	0.51	0.40
SimLex-999v1	0.32	0.17
MC30v2	0.72	0.58

Fig. 11: Correlación de Spearman entre el coseno de los *embeddings* y el puntaje de similaridad de la evaluación para cada par de palabras.

Pruebas de analogía con embeddings

Se realizaron múltiples pruebas de analogías utilizando los *embeddings* obtenidos por el sistema y por ELMo *for Many Langs*, en la Fig. 12 se reportan los resultados obtenidos.

	ELMo Entrenado	ELMo for Many Langs
Ciudad en estado	0.00	0.00
Nacionalidad	0.01	0.00
Moneda	0.00	0.00
Verbos en plural	0.22	0.09
Capitales del mundo	0.00	0.00
Plurales	0.01	0.02
Adjetivo a adverbio	0.03	0.04
Opuestos	0.00	0.00
Pasado	0.34	0.11
Gerundios	0.33	0.24
Familia	0.05	0.05
Capitales de países comunes	0.00	0.00

Fig. 12: Precisión alcanzada en las pruebas de analogía. En estos casos precisión y recall son idénticos.

Cabe señalar que estos resultados no muestran valores muy auspiciosos, pero es de esperarse que esto se deba a que se trata de los *embeddings* descontextualizados de las palabras (provenientes de la capa 0 de la red) los cuales no contienen una carga de información semántica significativa, ya que como señalan Peters et al. [2018], la información semántica se concentra en las capas superiores. De todas formas, los valores para el modelo entrenado parecen ser iguales o superiores a los del modelo de Che et al. [2018].

3.3. Desambiguador

El algoritmo global sigue entonces las siguientes etapas:

1. A cada instancia se le asocia una distribución de probabilidad sobre los posibles sustitutos dado su contexto.

2. A cada instancia se le asocian *k* representantes, cada uno conteniendo múltiples muestras de su distribución asociada (la distribución mencionada en el punto 1). Cada representante es un conjunto de tamaño 2*l,* conteniendo *l* muestras de la distribución *forward* y *l* muestras de la distribución *backward*.
3. Se clusterizan los representantes y se utiliza esta clusterización no ponderada de los representantes para derivar una clusterización ponderada de la instancia. Dadas *n* instancias de la palabra objetivo, se tienen entonces *nk* representantes en total, que se clusterizan en los distintos sentidos y se traduce esta clusterización de los representantes en una clusterización probabilística (ponderada) de las instancias originales.

Para la clusterización de los representantes se asocia cada representante con un vector de *bag-of-features* de dimensión |*V*|, donde *V* es el vocabulario de todos los representantes. Se tiene entonces matriz *M* de dimensión *nk x* |*V*|, donde cada fila es el vector de un representante. Las filas de *M* se clusterizan utilizando frecuencias relativas (*TF-IDF),* clusterización aglomerativa con distancia coseno y *average linkage*, y un número fijo de clusters (en las pruebas realizadas se fija el número máximo de clusters en 12, dado que es el máximo número de significados para una palabra en el diccionario de la tarea de evaluación).

A partir de la clusterización de los representantes se induce una clusterización ponderada de las instancias al asociar a cada instancia un valor de probabilidad de pertenencia al cluster determinado por la proporción de sus representantes que fueron asignados a dicho cluster.

Amrami y Goldberg utilizaron un modelo de lenguaje pre entrenado por Peters et al para su trabajo en el idioma inglés. Dado que al momento de la realización del presente trabajo no existía una versión entrenada de la red neuronal del modelo ELMo biLM disponible para el idioma español, se realizó la tarea de entrenar la red neuronal para español a partir de la implementación de la misma disponibilizada por el equipo de Peters, como se vio en la sección 3.2. En el proceso de desambiguación se utiliza el modelo entrenado, procesando lingüísticamente tanto su entrada como su salida, utilizando *Symmetric Patterns*, y lematizando las distribuciones de palabras resultantes.

Las tareas de WSI se definen sobre lemas, y algunas palabras objetivo tienen variantes morfológicas dentro de un mismo sentido (por ejemplo, los distintos tiempos verbales de un verbo utilizado con el mismo significado). Como los patrones simétricos favorecen a las palabras morfológicamente similares, los vectores sustitutos de dos instancias correspondientes a un mismo significado pero con diferencias en su forma, podrían diferir radicalmente, de acuerdo a la morfología (tiempo, género, número, etc.) de la instancia de la palabra objetivo. Para lidiar con este problema, las predicciones obtenidas del modelo de lenguaje son lematizadas antes de ser incluidas en los representantes.

Los autores afirman, tras experimentar con el modelo de lenguaje, que predicciones del modelo de lenguaje con valores altos de probabilidad usualmente son adecuadas al significado correcto de la instancia, pero los sustitutos con baja probabilidad aportan ruido a los representantes. Por lo tanto solo se consideran los 50 candidatos a sustituto con mayor

probabilidad para cada modelo de lenguaje (*forward* y *backward*), re-normalizando sus probabilidades correspondientes para que sumen 1.
Para la evaluación del método en inglés se utilizó la tarea 13 de SemEval 2013. Debido a la naturaleza estocástica del algoritmo, los resultados reportados por los autores corresponden a la media de la puntuación obtenida junto con su desviación estándar para 30 experimentos, y comparándolos con el estado del arte para dicha tarea al momento de publicación.
El código utilizado por Amrami y Goldberg [2018] se encuentra disponible en GitHub[20], y a fin de aplicarlo a la tarea Senseval 2- *Spanish Lexical Sample,* utilizando el modelo entrenado, se le debieron realizar los cambios que se describen en la siguiente subsección.
El siguiente diagrama (fig. 13) ilustra, a grandes rasgos, las interacciones entre los distintos componentes en el código modificado, a fin de facilitar más adelante la comprensión de las modificaciones realizadas.

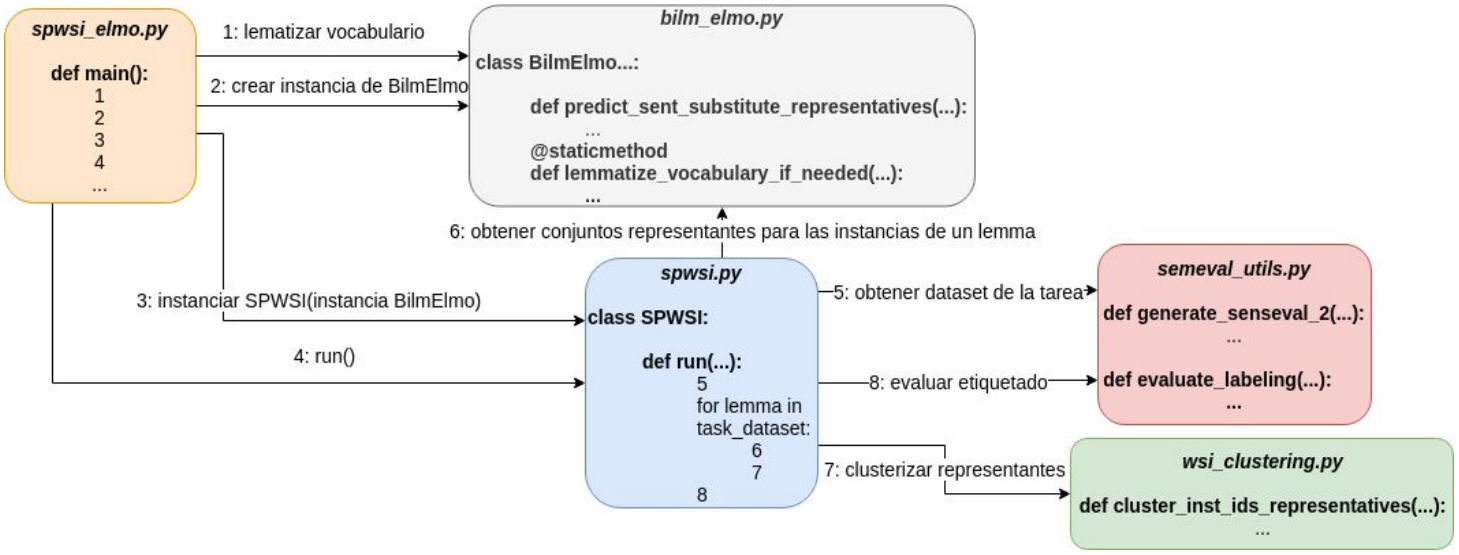

Fig. 13: Interacciones entre los diferentes componentes del desambiguador modificado.

[20] https://github.com/asafamr/SymPatternWSI

3.3.1. Modificaciones al desambiguador

spwsi_elmo.py

- Aquí se encuentra el *main* que ejecuta el proceso de desambiguación, se separó la sección de código que realiza efectivamente esa tarea en un método llamado *main()* del procesamiento de los parámetros recibidos por línea de comandos, con el fin de poder llamar al método de desambiguación desde otro módulo para realizar corridas repetidas del desambiguador desde un programa implementado para realizar 100 corridas consecutivas a fin de reportar el promedio y desviación estándar de las métricas.

- A los parámetros que recibe el programa se le agregaron los siguientes:

    - *--elmo-vocab-path*: parámetro que recibe la ruta al archivo de vocabulario del entrenamiento de biLM ELMo, el cual es también el vocabulario sobre el que se realizarán las predicciones.

    - *--weights-path*: parámetro que recibe la ruta del archivo que contiene el dump de los pesos obtenidos en el entrenamiento de biLM ELMo.

    - *--taskPath*: parámetro que recibe la ruta al directorio base conteniendo los archivos de la tarea Senseval 2 Spanish Lexical Sample.

    - *--maxLabels*: parámetro que indica el número de etiquetas que utilizará el desambiguador por instancia para evaluar sus resultados en la tarea.

    - *--options-path*: parámetro que recibe la ruta al archivo *options.json* con los parámetros utilizados para el entrenamiento de biLM ELMo.

  El valor *elmo-vocab-path* es pasado como parámetro al método *create_lemmatized_vocabulary_if_needed* de la clase *BilmElmo*. Los restantes parámetros mencionados son pasados al método *run* de la instancia de la clase *SPWSI*.

Bilm_elmo.py

La clase *BilmElmo* es la encargada de implementar el comportamiento del modelo de lenguaje.

- Se debió modificar la carga del modelo de la librería *Spacy* utilizada para lematizar el vocabulario, para que se cargara el modelo en español en lugar del correspondiente a inglés.

- En el método encargado de armar las oraciones objetivo para los modelos *forward* y *backward*, al utilizar *symmetric patterns* se reemplaza la conjunción copulativa del inglés, "*and*", por la del español, "y".

Semeval_utils.py

En este archivo se encuentra una variedad de funciones auxiliares para el manejo de los datos de la tarea a evaluar.

- Se implementaron funciones para realizar la carga de los datos de la tarea Senseval 2 Spanish Lexical Sample. La función *generate_senseval_2* implementa un generador que produce como salida una lista conteniendo las palabras de cada oración a desambiguar, la posición en la lista de la palabra a desambiguar en la lista, y un string conteniendo el lema de la palabra a desambiguar junto con su etiqueta de categoría sintáctica (*POS, part of speech*) en formato "*lemma.pos*".

- La función *evaluate_labeling* también requirió modificaciones. En primera instancia, se ha de resolver el problema de traducir las etiquetas producidas por el desambiguador, que corresponden simplemente a identificadores de los clusters construidos sin conocer el catálogo de significados de la tarea, a las etiquetas correspondientes al diccionario de esta. Para este propósito, se utiliza el mismo procedimiento de mapeo implementado por los autores de la tarea 13 de SemEval 2013, el cual se encuentra disponibilizadao por los autores de la tarea junto con el cálculo de las métricas que esta utiliza, disponible para su descarga en GitHub (https://github.com/alexanderpanchenko/cluster-comparison-tools). Dado que el código está implementado en Java, fue necesario utilizar el paquete PyJnius que permite interactuar desde Python con clases Java.

    - Como primer paso, es necesario obtener las respuestas correctas de la tarea. La función *getGoldKeySENSEVAL2* obtiene los datos de las respuestas de la tarea (el *gold standard*).

    - Luego, utilizando PyJnius, se implementaron las funciones auxiliares *dictToJ* que transforma un diccionario Python a un *HashMap* en Java, y *getTrainingInstances* que convierte los conjuntos de entrenamiento para el traductor de etiquetas de una lista en Python a un *HashSet* en Java.

    - La función auxiliar *mapSenses* instancia la clase *GradedReweightedKeyMapper* e invoca a su método *convert* pasando como

parámetros el diccionario con las respuestas de la tarea convertido a Java, el diccionario con las respuestas arrojadas por el desambiguador también convertido a Java y los conjuntos de entrenamiento construidos para el traductor de etiquetas. Luego la propia función auxiliar toma el resultado como objeto Java y lo transforma a un diccionario de Python.

- Con los datos de la *gold_key* y del mapeo de etiquetas resultante, se crea una instancia de la clase desarrollada *ad hoc Evaluator*, que interactúa mediante PyJnius con clases Java, también implementadas para este trabajo (las cuales están nombradas con el prefijo “Jnius” en el directorio *scoring*), que heredan de las clases específicas que implementan las métricas de WSD y WSI utilizadas en SemEval 2013 *Task* 13 en el código disponibilizado por los autores de la tarea.

- A su vez, la función interna *get_scores* pasa a ejecutar como subproceso en Python 2.7 el *scorer* original de la tarea Senseval 2 *Spanish Lexical Sample*, procesando luego su resultado devuelto como texto para cargar un diccionario.

Spwsi.py

La clase *SPWSI* carga el dataset de la tarea y se encarga de invocar a la clase que implementa el modelo de lenguaje. Luego realiza la clusterización de los representantes y llama al método *evaluate_labeling* de *semeval_utils* para obtener los resultados.

- Se modificó la carga del dataset en memoria, pasando a hacerse a través de la invocación a *generate_senseval_2* de *semeval_utils*.

- Se modificó la llamada a *evaluate_labeling* de *semeval_utils* para pasar como parámetro la ruta a la carpeta que contiene los datos de la tarea, el etiquetado realizado por la clusterización, el archivo donde se debe escribir la respuesta del sistema y el máximo de etiquetas por instancia a incluir en esta

4. Resultados

4.1. Líneas base

Para la comparación de los resultados del desambiguador se implementaron dos líneas base: una primera que asigna a cada instancia el primer significado para la palabra en el diccionario de la tarea (denominada *First Dictionary Sense, FDS*), y otra más compleja que utiliza los *embeddings* contextualizados de capa 2 de la red neuronal de biLM ELMo de las instancias a desambiguar (*Minimum Cosine Distance ELMo Layer* 2, *MCD-ELMo-L2*).

4.1.1. First Dictionary Sense (FDS)

Esta línea base simplemente asigna a cada instancia la etiqueta correspondiente al primer significado para la palabra en el diccionario de la tarea. Cabe notar que no se especifica si el primer significado fue seleccionado por ser el más frecuente según WordNet o cualquier otro catálogo de significados. Dada la simplicidad de esta línea base, no se requirió ninguna implementación específica, simplemente producir un archivo de respuestas con todas las instancias etiquetadas con “1” para ser evaluada por el scorer.

4.1.2. Minimum Cosine Distance ELMo Layer 2 (MCD-ELMo-L2)

Esta línea base obtiene el *embedding* contextualizado de capa 2 de cada instancia a desambiguar y la compara con el centroide de los *embeddings* de la definición y de los ejemplos de uso provistos por el diccionario de la tarea para la palabra. Aquel que resulte con la menor distancia coseno, es seleccionado como el significado correcto para la instancia.
Se utiliza la capa 2 de la red neuronal dado que según sostienen los autores de ELMo [Peters et al., 2018] es donde se concentra la mayor cantidad de información semántica.
La implementación de esta línea base se encuentra en el archivo *embedding_baseline.py*.

4.2. Análisis de log del desambiguador

Como ejemplo, tomemos del log de una corrida del desambiguador la salida correspondiente a la instancia *actuar.000001*:

*“De casta le viene al galgo (en este caso a la galga) : la madre de la protagonista de Shakespeare enamorado - - la actriz teatral Blythe Danner , que también suele **actuar** en películas de Woody Allen - - interpretaba una obra del genial dramaturgo inglés estando embarazada de Gwyneth .”*

El log refleja que algunos posibles sustitutos obtenidos del modelo de lenguaje *forward* son *“cantar”, “representar”, “educar”, “explotar”, “compartir”* y en el caso del modelo *backward* se obtienen *“actor”, “televisión”, “interpretar”, “tocar”*.
Considerando el uso de *Symmetric Patterns*, esto implicaría que el modelo *forward* propone completar la primera parte de la siguiente manera:

“De casta le viene al galgo (en este caso a la galga) : la madre de la protagonista de Shakespeare enamorado - - la actriz teatral Blythe Danner , que también suele actuar y [cantar | representar | educar | explotar | compartir | ...]”

Mientras que el *backward* completa la segunda mitad de la siguiente manera:

“[actor | televisión | interpretar | tocar | …] y actuar en películas de Woody Allen - - interpretaba una obra del genial dramaturgo inglés estando embarazada de Gwyneth .”

Con estos candidatos a sustitutos, tomados de ambos modelos de lenguaje se forman representantes como los siguientes ejemplos también tomados del log:

{'expresar': 1, 'bailarín': 1, 'cantar': 1, 'estudiar': 1, 'transmitir': 1, 'comediar': 1, 'representar': 1, 'cine': 1}

{'representar': 1, 'cantar': 2, 'prestar': 1, 'interpretar': 2, 'difundir': 2}

{'educar': 1, 'Broadway': 1, 'convertirse': 1, 'cortometraje': 1, 'prestar': 1, 'escribir': 1, 'presidir': 1, 'cantar': 1}

Donde el número que sigue a la palabra indica la multiplicidad de veces que la misma resultó del proceso de muestreo ponderado utilizado para formar el conjunto representante.
Como puede apreciarse, algunas de las palabras sugeridas por los modelos son sustitutos muy acertados en el contexto de la oración, mientras que otras no lo son (incluso sin pertenecer a la categoría gramatical que sería esperable, en este caso: un verbo) pero son conceptos muy relacionados a la temática de la misma.

4.3. Resultados experimentales

Como se ha mencionado, el método de desambiguación se evalúa sobre la tarea Senseval 2- *Spanish Lexical Sample*, consistente en 39 palabras ambiguas con un promedio de 57 instancias por palabra, donde cada instancia está anotada con entre una y dos etiquetas de un diccionario construido *ad hoc* para la tarea.
En todos los casos, los valores numéricos reportados correponden al valor de la métrica normalizado a [0,1] multiplicado por 100 para facilitar su comparación.

Para facilitar la comprensión de los cuadros de resultados, la Fig. 14 detalla los modelos evaluados:

Nombre corto	Detalle
SP-ELMO-1L-12C-20R-4S-L-TFIDF	*Symmetric Patterns* de Amrami y Goldberg [2018] con 1 etiqueta por instancia, 12 clusters como máximo y 20 representantes formados por 4 muestras por LM (*forward* y *backward*), con lematización, aplicando TF-IDF a la matriz de coocurrencias
noSP-ELMO-1L-12C-20R-4S-L-TFIDF	Idem anterior, sin utilizar *symmetric patterns*
SP-ELMO-1L-12C-20R-4S-TFIDF	Idem al modelo base, sin utilizar lematización
SP-ELMO-1L-12C-20R-4S-L	Idem al modelo base, sin aplicar TF-IDF a la matriz de coocurrencias previo a la clusterización
SP-ELMO-2L-12C-20R-4S-L-TFIDF	*Symmetric Patterns* de Amrami y Goldberg [2018] con 2 etiquetas por instancia, 12 clusters como máximo y 20 representantes formados por 4 muestras por LM (*forward* y *backward*), con lematización, aplicando TF-IDF a la matriz de coocurrencias
noSP-ELMO-2L-12C-20R-4S-L-TFIDF	Idem anterior, sin utilizar *symmetric patterns*
SP-ELMO-2L-12C-20R-4S-TFIDF	Idem al modelo base, sin utilizar lematización, con 2 etiquetas por instancia
SP-ELMO-2L-12C-20R-4S-L	Idem al modelo base, sin aplicar TF-IDF a la matriz de coocurrencias previo a la clusterización, con 2 etiquetas por instancia
MCD-ELMO-L2	Línea base: Mínima distancia coseno entre ELMo capa 2 de target y centroide de ELMo capa 2 de definiciones y ejemplos de uso de posibles sentidos tomados del diccionario de la tarea
FDS	Línea base: Primer significado en el diccionario de la tarea

Fig. 14: Descripción de los sistemas evaluados.

Cabe destacar que una amplia mayoría de las instancias tienen como respuesta correcta una única etiqueta, existiendo 29 instancias (de un total de 2225) etiquetadas con 2 etiquetas. Para estos casos, se consideró la aplicabilidad de cada etiqueta como 0.5.
Dada la naturaleza estocástica del algoritmo, los resultados reportados corresponden a los promedios de 100 ejecuciones, reportándose el promedio junto con la desviación estándar.
Se consideran tres marcos de evaluación, uno consistente en las métricas correspondientes a la tarea Senseval 2- *Spanish Lexical Sample* que permite evaluar el algoritmo en comparación a otros sobre una misma tarea. Luego se consideran las métricas propuestas para la tarea SemEval 2013 *Task 13*, en WSD y finalmente en WSI. Los cálculos de las métricas se realizaron utilizando el software de *scoring* disponibilizado por las respectivas tareas (implementando adaptadores que permitan invocarlos desde Python en el caso del software de evaluación de SemEval *Task* 13, dado que se encuentran implementados en Java).
La comparación contra los resultados reportados para la tarea Senseval 2- *Spanish Lexical Sample*, ubican al método evaluado en un virtual cuarto lugar, mostrando resultados comparables al de métodos basados en aprendizaje supervisado, lo cual resulta altamente positivo para un método no supervisado.

4.3.1. Resultados de métricas de Senseval 2- Spanish Lexical Sample

En la Fig. 15 se presenta el resultado obtenido para cada modelo evaluado en la tarea SensEval 2- Spanish Lexical Sample, comparado contra las líneas base definidas:

	Precisión	Recall	F-Measure	Respuestas correctas	Respuestas intentadas
SP-ELMO-1L-12C-20R-4S-L-TFIDF	64.70±0.85	**64.29±0.85**	**64.49±0.85**	**1430.34±18.94**	2210.52±3.49 (99.35±0.15%)
noSP-ELMO-1L-12C-20R-4S-L-TFIDF	**65.77±0.97**	62.96±0.93	64.33±0.95	1400.68±20.84	2129.78±7.44 (95.72±0.33%)
SP-ELMO-1L-12C-20R-4S-L	62.05±0.74	61.84±0.74	61.94±0.74	1375.81±16.42	2217.23±2.01 (99.65±0.09%)
SP-ELMO-1L-12C-20R-4S-TFIDF	62.37±0.74	60.83±0.75	61.59±0.75	1353.41±16.56	2169.97±4.86 (97.53±0.22%)
noSP-ELMO-2L-12C-20R-4S-L-TFIDF	45.47±0.52	43.53±0.51	44.48±0.51	968.66±11.28	2130.65±6.76 (95.76±0.30%)
MCD-ELMO-L2	43.4	43.4	43.4	966	2225 (100%)
SP-ELMO-2L-12C-20R-4S-L-TFIDF	43.21±0.29	42.90±0.32	43.06±0.30	954.62±6.87	2209.47±3.49 (99.30±0.16%)
SP-ELMO-2L-12C-20R-4S-TFIDF	43.04±0.33	41.96±0.33	42.49±0.33	933.68±7.45	2169.61±5.13 (97.51±0.23)
SP-ELMO-2L-12C-20R-4S-L	42.23±0.28	42.08±0.28	42.15±0.28	936.34±6.26	2217.57±1.95 (99.67±0.09%)
FDS	38.7	38.7	38.7	861.5	2225 (100%)

Fig. 15: Resultados obtenidos por el modelo evaluado y las líneas base consideradas en la tarea Senseval2- *Spanish Lexical Sample*

El sistema que obtiene el mayor nivel de *F1* es el que utiliza el método completo sin ablaciones, aunque puede verse que el sistema que no aplica *symmetric patterns* alcanza una mayor precisión. Las variantes que utilizan hasta 2 etiquetas por instancia, tienen un rendimiento notoriamente menor a los que utilizan 1 única etiqueta.

Por tanto, de acuerdo a los resultados reportados en su momento en la tarea Senseval 2- *Spanish Lexical Sample* (Fig. 16), puede verse que el método implementado, construido únicamente a partir de texto sin anotar, obtiene unos resultados competitivos con los de sistemas supervisados que utilizan recursos lexicográficos sofisticados, ubicándose en un virtual cuarto lugar:

	Supervisado	Precisión	Recall	F1
JHU(R)	Si	**71.2**	**71.2**	**71.2**
JHU	Si	68.1	68.1	68.1
Stanford- CS224N	Si	66.9	66.9	66.9
SP-ELMO-1L-12C-20R-4S-L-TFIDF	**No**	64.70±0.85	64.29±0.85	64.49±0.85
UMD-SST	Si	62.7	62.7	62.7
Duluth 8	Si	61.5	61.5	61.5

Fig. 16: Resultados reportados para la tarea Senseval 2- *Spanish Lexical Sample*, con el agregado de los resultados obtenidos para el modelo analizado.

4.3.2. Resultados de métricas de SemEval 2013 Task 13 en WSD

En la Fig. 17 se muestran los valores alcanzados en las métricas definidas para la tarea SemEval 2013 *Task* 13, en este caso aplicadas a las respuestas obtenidas para la tarea Senseval 2 *Spanish Lexical Sample:*

	Jaccard Index	Positional Kendall's Tau	WNDCG
SP-ELMO-1L-12C-20R-4S-L-TFIDF	64.21±0.85	64.03±0.86	33.82±0.55
noSP-ELMO-1L-12C-20R-4S-L-TFIDF	**65.27±0.96**	65.08±0.98	37.93±0.70
SP-ELMO-1L-12C-20R-4S-TFIDF	61.87±0.75	61.67±0.75	33.43±0.56
SP-ELMO-1L-12C-20R-4S-L	61.57±0.74	61.36±0.75	30.34±0.57
SP-ELMO-2L-12C-20R-4S-L-TFIDF	43.12±0.30	66.55±0.91	36.48±0.49
noSP-ELMO-2L-12C-20R-4S-L-TFIDF	45.35±0.52	**68.01±0.82**	**40.22±0.62**
SP-ELMO-2L-12C-20R-4S-TFIDF	42.95±0.33	64.55±0.74	36.10±0.45
SP-ELMO-2L-12C-20R-4S-L	42.14±0.28	64.44±0.77	33.45±0.43
MCD-ELMO-L2	42.88	42.99	31.97
FDS	38.26	37.97	28.72

Fig. 17: Resultados de métricas de WSD propuestas por SemEval 2013 Task 13 para los sistemas evaluados en Senseval 2- *Spanish Lexical Sample*

Como puede verse en la Fig. 17, la variante sin *symmetric patterns* utilizando una etiqueta obtiene el mejor valor de Jaccard Index, sin embargo es superada en *Positional Kendall's Tau* y WNDCG por la variante con 2 etiquetas también sin *symmetric patterns*.

4.3.3. Resultados de métricas de SemEval 2013 Task 13 en WSI

En la Fig. 18 se presentan los resultados obtenidos por el sistema analizado y las líneas base propuestas en las métricas presentadas por SemEval 2013 Task 13, calculadas sobre la tarea Senseval 2- *Spanish Lexical Sample.*

	FBC Precision	FBC- Recall	FBC- F1	FNMI
SP-ELMO-1L-12C-20R-4S-L-TFIDF	57.54±0.92	49.03±0.75	52.94±0.92	20.50±0.69
noSP-ELMO-1L-12C-20R-4S-L-TFIDF	53.43±1.03	51.12±1.01	52.25±1.02	**22.70±1.04**
SP-ELMO-1L-12C-20R-4S-TFIDF	52.85±0.83	47.04±0.73	49.77±0.78	18.78±0.72
SP-ELMO-1L-12C-20R-4S-L	61.88±0.86	46.33±0.75	52.99±0.80	18.09±0.73
SP-ELMO-2L-12C-20R-4S-L-TFIDF	83.82±0.57	27.79±0.34	41.74±0.43	16.55±0.42
noSP-ELMO-2L-12C-20R-4S-L-TFIDF	75.03±0.70	32.08±0.54	44.95±0.61	19.25±0.81
SP-ELMO-2L-12C-20R-4S-TFIDF	78.02±0.49	29.34±0.34	42.65±0.40	15.56±0.51
SP-ELMO-2L-12C-20R-4S-L	87.72±0.51	26.00±0.28	40.11±0.36	14.98.0.48
MCD-ELMO-L2	59.26	45.84	49.94	14.29
FDS	98.94	37.69	**53.44**	5.80

Fig. 18: Resultados de las métricas de WSI propuestas por SemEval 2013 Task 13 para los sistemas evaluados en Senseval 2- *Spanish Lexical Sample*

En la fig. 18, puede verse que la línea base del primer significado del diccionario (*FDS*) supera a todas las variantes del método en FBC F1. Este resultado se explica por el hecho de que puede verse *Fuzzy B-Cubed* como una medida del ajuste entre dos clusterizaciones a nivel de ítems. Para un item *i*, la precisión refleja cuántos items de los que comparten cluster con *i* en la primer clusterizacion, también se encuentran en un cluster junto con *i* en la segunda clusterización, y el *recall* mide cuántos items que comparten cluster en la segunda clusterización con *i* también lo comparten en la primera [Jurgens y Klapaftis, 2013]. Dado que en la línea base FDS todos los ítems comparten el mismo cluster en la clusterización correspondiente a la respuesta del sistema, para cualquier par de elementos que compartan cluster en la clusterización *gold standard* también lo compartirán en la

respuesta del sistema, dando como resultado un valor muy alto de *FBC Precision*. A la inversa, será menos frecuente que elementos que compartan cluster en la respuesta del sistema compartan cluster en el *gold standard*, el FBC *Recall* se ve resentido. Sin embargo el efecto de esta baja en el *recall* no alcanza a compensar el alto valor de la precisión (probablemente debido a que las instancias en el corpus de la tarea no se distribuyen uniformemente entre los sentidos) dando un valor de FBC F1 superior al del método evaluado. Esto debe interpretarse como un mejor ajuste de *FDS* a corpus que tengan una distribución de las instancias en los significados similar a la de la tarea.
El mayor FNMI es alcanzado por la variante con una etiqueta y sin *symmetric patterns*. Las variantes con 2 etiquetas tienen un menor rendimiento en FBC F1, pero alcanzan valores similares a los sistemas que utilizan una única etiqueta para FNMI.
Otra comparación posible resulta de contrastar estos valores obtenidos con los reportados por Amrami y Goldberg [2018] para la tarea SemEval 2013 *Task* 13. En dicha tarea el sistema obtuvo un FNMI de 11.26 y un FBC de 57.49. Como puede verse, para el español el FBC alcanzado es similar, aunque el FNMI resulta notoriamente mayor, llegando a alcanzar el doble del reportado para el inglés. De todas formas, esta comparación debe tomarse con extremo cuidado, ya que son resultados obtenidos para idiomas diferentes, sobre tareas diferentes.

5. Conclusiones y trabajo futuro

En el desarrollo de este proyecto se obtuvieron y compararon resultados de WSD y WSI para la técnica de WSI con modelos de lenguaje neuronales y *symmetric patterns* de Amrami y Goldberg [2018] para el idioma español, sobre la tarea Senseval 2 *Spanish Lexical Sample.*

Se midió el rendimiento de 4 configuraciones distintas del método de desambiguación con 1 etiqueta por instancia: una versión completa que utiliza *symmetric patterns*, lematización y TF-IDF para la matriz de coocurrencias previo a la clusterización, y luego una versión con cada una de esas 3 características desactivadas, para luego considerar cada una de estas variantes utilizando un máximo de 2 etiquetas por instancia.

El método en todas las variantes analizadas con 1 etiqueta aplicado a la resolución del problema de WSI en español arroja resultados competitivos para la tarea elegida como evaluación, aún comparándola contra sistemas supervisados en un contexto de WSD. Es de destacar que mientras estos sistemas se basan en recursos lingüísticos construidos a partir del trabajo de expertos humanos, el sistema analizado alcanza un rendimiento comparable partiendo únicamente de una gran colección de texto sin anotar, utilizado para entrenar el modelo de lenguaje.

Los resultados en WSI son también aproximadamente similares a los reportados para el inglés, aunque esta comparación es poco relevante dado que no se puede hacer una comparación directa entre métricas obtenidas en distintas tareas para distintos idiomas.

Contrariamente a lo expuesto por Amrami y Goldberg [2018] para el inglés, la variante que no aplica *symmetric patterns* obtuvo un mejor rendimiento en las medidas de WSI que aquellas que sí lo aplican. Es prematuro alcanzar conclusiones sobre el motivo de esta observación, que podría deberse desde particularidades en el corpus de evaluación de la tarea hasta a características lingüísticas propias del español que lo diferencien del inglés.

Por otro lado, puede observarse que la caída más importante en el rendimiento del método en WSI se da al desactivar la aplicación de frecuencias relativas a la matriz de coocurrencias previo a la clusterización, en contraste con el inglés donde los autores reportaron la mayor baja en el rendimiento del método al desactivar la lematización.

Los resultados del desambiguador también superaron a las líneas base construidas en la evaluación WSD de la tarea Senseval 2- *Spanish Lexical Sample*, y en *FNMI* para WSI, siendo superado por la línea base del primer sentido del diccionario en *FBC*. Sin embargo, como se ha expuesto, esto solo significaría un mejor ajuste de *FDS* a un corpus con una distribución de los significados en las instancias similar al del corpus de evaluación.

Los resultados de este trabajo podrán a su vez ser usados como referencia para la comparación de otros sistemas de WSI en español, algo que resulta relevante dado que al momento de realización del presente trabajo no es posible obtener resultados reportados de WSI en este idioma.

Para poder aplicar la técnica de WSI con modelos de lenguaje neuronales y *symmetric patterns* se realizó el entrenamiento de un modelo de lenguaje basado en una red neuronal biLM ELMo en español. De este entrenamiento resulta no solo en un modelo de lenguaje

neuronal completo que podrá ser utilizado en el futuro como tal, si no también para la obtención de *embeddings* contextualizados (o no contextualizados). La evaluación realizada de los *embeddings* fuera de contexto, muestra que tal como en el inglés, estos no parecen aportar suficiente información semántica. Una evaluación más profunda de la calidad de los *embeddings* contextualizados excede el alcance de este trabajo, aunque resulta un interesante objetivo para futuras investigaciones, así como la aplicación de los *embeddings* a distintas tareas de PLN.

Como observación para buscar enriquecer los recursos para el desarrollo de la investigación del problema de WSI en español, sería algo por demás importante contar con una tarea de evaluación similar a la tarea SemEval 2013 *Task* 13 para el español, de forma que diferentes sistemas puedan compararse de forma más directa y estandarizada.

6. Referencias

- Agirre, E., & Soroa, A. (2009, March). Personalizing pagerank for word sense disambiguation. In Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics (pp. 33-41). Association for Computational Linguistics.
- Agirre, E., De Lacalle, O. L., & Soroa, A. (2009, June). Knowledge-based WSD on specific domains: performing better than generic supervised WSD. In Twenty-First International Joint Conference on Artificial Intelligence.
- Agirre, E., López de Lacalle, O., & Soroa, A. (2014). Random walks for knowledge-based word sense disambiguation. Computational Linguistics, 40(1), 57-84.
- Agirre, E., Martínez, D., De Lacalle, O. L., & Soroa, A. (2006, June). Evaluating and optimizing the parameters of an unsupervised graph-based WSD algorithm. In Proceedings of the first workshop on graph based methods for natural language processing (pp. 89-96). Association for Computational Linguistics.
- Alpaydin, E. (2009). Introduction to machine learning. MIT press.
- Amigó, E., Gonzalo, J., Artiles, J., & Verdejo, F. (2009). A comparison of extrinsic clustering evaluation metrics based on formal constraints. Information retrieval, 12(4), 461-486.
- Amrami, A., & Goldberg, Y. (2018). Word Sense Induction with Neural biLM and Symmetric Patterns. arXiv preprint arXiv:1808.08518.
- Baskaya, O., Sert, E., Cirik, V., & Yuret, D. (2013, June). Ai-ku: Using substitute vectors and co-occurrence modeling for word sense induction and disambiguation. In Second Joint Conference on Lexical and Computational Semantics (* SEM), Volume 2: Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval 2013) (pp. 300-306).
- Bengio, Y., Ducharme, R., Vincent, P., & Jauvin, C. (2003). A neural probabilistic language model. Journal of machine learning research, 3(Feb), 1137-1155.
- Brody, S., & Lapata, M. (2009, March). Bayesian word sense induction. In Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics (pp. 103-111). Association for Computational Linguistics.
- Cardellino, C. (2016). Spanish billion words corpus and embeddings. Consultado el 9 de septiembre de 2019. Sitio web: https://crscardellino.github.io/SBWCE/

- Che, W., Liu, Y., Wang, Y., Zheng, B., & Liu, T. (2018). Towards better UD parsing: Deep contextualized word embeddings, ensemble, and treebank concatenation. arXiv preprint arXiv:1807.03121.
- Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805.
- Firth, J. R. (1961). Papers in Linguistics 1934-1951: Repr. Oxford University Press.
- Garousi, V., Bauer, S., & Felderer, M. (2018). NLP-assisted software testing: a systematic review. arXiv preprint arXiv:1806.00696.
- Gers, F. (2001). Long short-term memory in recurrent neural networks (Doctoral dissertation).
- Gurney, K. (2014). An introduction to neural networks. CRC press.
- Gutiérrez, Y., Fernández Orquín, A., Montoyo, A., & Vázquez, S. (2011). Enriching the integration of semantic resources based on wordnet.
- Gutiérrez, Y. (2012). Análisis semántico multidimensional aplicado a la desambiguación del lenguaje natural.
- Hagiwara, M.. (2018). Improving a Sentiment Analyzer using ELMo — Word Embeddings on Steroids. Consultado el 21 de septiembre de 2019. Sitio web: http://www.realworldnlpbook.com/blog/improving-sentiment-analyzer-using-elmo.html
- Ide, N., & Véronis, J. (1998). Introduction to the special issue on word sense disambiguation: the state of the art. Computational linguistics, 24(1), 2-40.
- Jurgens, D., & Klapaftis, I. (2013, June). Semeval-2013 task 13: Word sense induction for graded and non-graded senses. In Second Joint Conference on Lexical and Computational Semantics (* SEM), Volume 2: Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval 2013) (pp. 290-299).
- Jurgens, D. (2012, June). An evaluation of graded sense disambiguation using word sense induction. In Proceedings of the First Joint Conference on Lexical and Computational Semantics-Volume 1: Proceedings of the main conference and the shared task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation (pp. 189-198). Association for Computational Linguistics.
- Komninos, A., & Manandhar, S. (2016, December). Structured generative models of continuous features for word sense induction. In Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers (pp. 3577-3587).

- Kurdi, M. Z. (2016). Natural language processing and computational linguistics: speech, morphology and syntax (Vol. 1). John Wiley & Sons.
- Levy, O., & Goldberg, Y. (2014). Neural word embedding as implicit matrix factorization. In Advances in neural information processing systems (pp. 2177-2185).
- Manandhar, S., Klapaftis, I. P., Dligach, D., & Pradhan, S. S. (2010, July). SemEval-2010 task 14: Word sense induction & disambiguation. In Proceedings of the 5th international workshop on semantic evaluation (pp. 63-68). Association for Computational Linguistics.
- Mandic, D. P., & Chambers, J. (2001). Recurrent neural networks for prediction: learning algorithms, architectures and stability. John Wiley & Sons, Inc.
- Manion, S. L., & Sainudiin, R. (2014). An iterative 'sudoku style' approach to subgraph-based word sense disambiguation. In Proceedings of the Third Joint Conference on Lexical and Computational Semantics (* SEM 2014) (pp. 40-50).
- Maron, Y., Lamar, M., & Bienenstock, E. (2010). Sphere embedding: An application to part-of-speech induction. In Advances in Neural Information Processing Systems (pp. 1567-1575).
- Màrquez, L., Taulé, M., Martí, A., Artigas, N., García, M., Real, F., & Ferrés, D. (2004). Senseval-3: The Spanish lexical sample task. In Proceedings of SENSEVAL-3, the Third International Workshop on the Evaluation of Systems for the Semantic Analysis of Text (pp. 21-24).
- Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781.
- Miller, G. A., Beckwith, R., Fellbaum, C., Gross, D., & Miller, K. J. (1990). Introduction to WordNet: An on-line lexical database. International journal of lexicography, 3(4), 235-244.
- Miller, G. A. (1990). Nouns in WordNet: a lexical inheritance system. International journal of Lexicography, 3(4), 245-264.
- Moro, A., & Navigli, R. (2015, June). Semeval-2015 task 13: Multilingual all-words sense disambiguation and entity linking. In Proceedings of the 9th international workshop on semantic evaluation (SemEval 2015) (pp. 288-297).
- Navigli, R., & Ponzetto, S. P. (2010, July). BabelNet: Building a very large multilingual semantic network. In Proceedings of the 48th annual meeting of the association for computational linguistics (pp. 216-225). Association for Computational Linguistics.
- Navigli, R., Jurgens, D., & Vannella, D. (2013, June). Semeval-2013 task 12: Multilingual word sense disambiguation. In Second Joint Conference on Lexical and

Computational Semantics (* SEM), Volume 2: Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval 2013) (pp. 222-231).

- Navigli, R. (2009). Word sense disambiguation: A survey. ACM computing surveys (CSUR), 41(2), 10.
- Navigli, R. (2012, January). A quick tour of word sense disambiguation, induction and related approaches. In International Conference on Current Trends in Theory and Practice of Computer Science (pp. 115-129). Springer, Berlin, Heidelberg.
- Oakes, M.& McCarthy, D.. (2002). SENSEVAL II System Descriptions. Consultado el 8 de septiembre de 2019. Sitio web: http://www.dianamccarthy.co.uk/SEVALsystems.html
- Pantel, P., & Lin, D. (2002, July). Discovering word senses from text. In Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining (pp. 613-619). ACM.
- Peters, M. E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., & Zettlemoyer, L. (2018). Deep contextualized word representations. arXiv preprint arXiv:1802.05365.
- Russell, S. J., & Norvig, P. (2016). Artificial intelligence: a modern approach. Malaysia; Pearson Education Limited,.
- Schwab, D., Goulian, J., Tchechmedjiev, A., & Blanchon, H. (2012, December). Ant colony algorithm for the unsupervised word sense disambiguation of texts: Comparison and evaluation. In Proceedings of COLING 2012 (pp. 2389-2404).
- Van de Cruys, T., & Apidianaki, M. (2011, June). Latent semantic word sense induction and disambiguation. In Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1 (pp. 1476-1485). Association for Computational Linguistics.
- Yampolskiy, R. V. (2013). Turing test as a defining feature of AI-completeness. In Artificial intelligence, evolutionary computing and metaheuristics (pp. 3-17). Springer, Berlin, Heidelberg.

Glosario

LSTM: Sigla en inglés de *Long Short Term Memory*. Tipo de red neuronal recurrente caracterizada por utilizar *memory blocks,* más complejos que las neuronas artificiales típicas.

NLP: Sigla en inglés de *Natural Language Processing*. Procesamiento de lenguaje natural.

PLN: Procesamiento de lenguaje natural.

POS: Sigla en inglés de *Part of Speech.* Partes de la oración, por ej: sustantivo, verbo, adjetivo, pronombre, etc.

RNN: Sigla en inglés de *Recurrent Neural Network.* Red neuronal recurrente.

WSD: Sigla en inglés de *Word Sense Disambiguation*, problema de procesamiento de lenguaje natural donde se busca asignar significados de un inventario dado a instancias de las palabras objetivo.

WSI: Sigla en inglés de *Word Sense Induction*, problema de procesamiento de lenguaje natural donde se busca descubrir agrupamientos de instancias de una palabra objetivo que comparten un significado.

A. Anexo: Sistemas participantes en las tareas de WSD en español

A.1. Senseval 2- Spanish Lexical Sample Task[21]

JHU

Consiste en 6 subsistemas de aprendizaje supervisado integrados mediante un clasificador Los subsistemas incluyen listas de decisión, aprendizaje basado en transformaciones, modelos vectoriales y sistemas basados en *naive* Bayes.

CS224N

Formado por una colección de clasificadores de significados, utilizando *naive* Bayes, modelos vectoriales, n-gramas y k-vecinos más próximos, implementando una variedad de técnicas de construcción de ventanas de palabras, ponderación de distancias y técnicas de suavizado. Estos clasificadores de primer nivel son combinados por un clasificador de segundo nivel, que puede usar votación simple, votación ponderada o un modelo Log-lineal.

UMD-SST

Este método utiliza aprendizaje mediante *support vector machines* (SVM) para construir clasificadores a partir de los datos de entrenamiento. Los vectores de co-ocurrencia ponderados por IDF contienen información de co-ocurrencias tanto locales como lejanas.

Duluth-6

Este sistema induce tres árboles de decisión a partir de ejemplos etiquetados. Cuando se presenta un ejemplo a desambiguar, cada árbol de decisión da como salida probabilidades para cada significado posible, las cuales se suman y el sentido con el máximo valor es seleccionado. No se utiliza información de WordNet.

[21] [Oakes y McCarthy, 2002]

A.2. SemEval 2013 Task 12: Multilingual Word Sense DIsambiguation

DAEBAK!

Utiliza una ventana de 5 palabras alrededor de la palabra objetivo, construye un grafo para todos los significados de los lemas contenidos en la ventana, basándose en BabelNet. Luego elige el significado para etiquetar la instancia midiendo la conectividad con los *synsets* de los lemas cercanos.

GETALP

Este equipo participó con 3 sistemas , dos basados en BabelNet y uno en WordNet, basados en el algoritmo *ant-colony* de Schwab et al. [2012] que utiliza la estructura de red del inventario de significados para identificar caminos que conectan *synsets* del lema objetivo con los *synsets* otros lemas en el contexto. El sistema BN1 optimiza sus parámetros a partir del corpus de entrenamiento, mientras que el BN2 y el WN1 son completamente no supervisados y optimizan sus parámetros directamente a partir de la estructura de grafo de BabelNet y WordNet respectivamente. Sólo BN2 fue presentado para el idioma español.

UMCC-DLSI

UMCC-DLSI presentó tres sistemas basados en ISR-WN de Gutiérrez et al. [2011] que enriquece la red semántica de WordNet utilizando aristas de varios recursos léxicos. Luego la desambiguación se realiza utilizando la red construida con el algoritmo presentado por Gutiérrez [2012] que extiende el algoritmo de WSD de *Personalized Page Rank* de Agirre y Soroa [2009] que incluye información sobre la frecuencia del significado. Sólo los sistemas *Run-1* y *Run-2* se presentaron para la evaluación en español. El sistema *Run-1* realiza WSD usando todas las instancias de sustantivos en el contexto de la oración, mientras que *Run-2* funciona a nivel de discurso e inicializa *PageRank* utilizando los *synsets* de todos los sustantivos en el documento.

A.3. SemEval 2015 Task 13: Multilingual All-Words Sense Disambiguation and Entity Linking

LIMSI

Implementa un algoritmo no supervisado que incorpora elementos de relevancia de significados. Los textos en cada uno de los diferentes idiomas son alineados a nivel de oración y de palabra de a pares, y las palabras son etiquetadas por sus traducciones en los otros idiomas. Los alineamientos luego se utilizan para obtener los conjuntos de sinónimos de BabelNet relevantes para cada instancia de la palabra en los textos (por ejemplo

aquellos conjuntos de sinónimos que contienen tanto la palabra a desambiguar como sus traducciones alineadas).

SUDOKU

Se trata de un método no supervisado basado en *Personalized Page Rank* de Agirre et al. [2014], donde este se inicia con un vector sesgado hacia palabras monosémicas. Presenta 3 variantes: *Run 1* consta del método base propuesto por Manion y Sainudiin [2014] aplicado a nivel del documento, *Run 2* es la versión iterativa de la variante anterior aplicada a nivel de documento y con las palabras desambiguadas en orden creciente de polisemia, *Run3* es análogo a *Run 2*, pero primero aplicado primero a sustantivos, y luego a verbos, adjetivos y adverbios.

B. Anexo: Métricas de evaluación en WSI de SemEval 2013: Task 13

B.1. Fuzzy B-Cubed

Para la definición de *Fuzzy B-Cubed*, siguiendo la formalización de Amigó et al. [2009] donde se tiene una definición de funciones de precisión y *recall P* y *R* para items, se define una función *avg* que retorna el valor promedio de una serie y sea $\mu_x(i)$ el conjunto de clusters en el clustering *X* de los cuales el elemento *i* es miembro. La precisión y el *recall* de B-Cubed se calculan sobre los *n* items de la siguiente forma:

$$BCubed\ Precision = avg_i\ [avg_{j \neq i \in \cup \mu_y(i)}\ P(i,j)]$$

$$BCubed\ Recall = avg_i[avg_{j \neq i \in \cup \mu_x(i)} R(i,j)]$$

Donde *P* y *R* se definen como 1 si las etiquetas de cluster de dos ítems son idénticas. Sea $l_X(i)$ el conjunto de clusters de los cuales es miembro *i*, y $w_k(i)$ el grado de aplicabilidad de la pertenencia del item *i* al cluster *k* en *X*. Se define *C* respecto a *X* como:

$$C(i,j,X) = \sum_{k \in l_X(i) \cup l_X(j)} 1 - \left| w_k(i) - w_k(j) \right|.$$

Puede verse que *C* se maximiza cuando *i* y *j* tienen aplicabilidad idéntica en los clusters de los cuales son miembros. La precisión y *recall* para un par de items se definen en términos de *C* como $P(i,j,X) = \frac{Min(C(i,j,X),C(i,j,Y))}{C(i,j,X)}$ y $R(i,j,X) = \frac{Min(C(i,j,X),C(i,j,Y))}{C(i,j,Y)}$ respectivamente, las que son luego utilizadas para el cálculo de *BCubed Precision* y *BCubed Recall*.

Supongamos que tenemos 3 instancias (1, 2 y 3) de una misma palabra a desambiguar, clusterizadas de forma ponderada por el sistema a evaluar en la clusterización X, y la clusterización ponderada *gold standard* Y, como se ve en la fig. 19:

	X	Y
1	A: 0.6, B: 0.3, C:0.1	A: 0.7, C: 0.2, D:0.1
2	B: 0.5, A: 0.4, C: 0.1	D: 0.6, B: 0.4
3	A: 0.7, C: 0.3	C: 0.6, A: 0.4

Fig. 19: Ejemplo de etiquetado.

Entonces para el cálculo del *recall*, consideramos para la instancia 1 los clusters que le corresponden en Y: A, C y D. Puede verse que en Y, comparte el cluster D con 2, para el cual 1 tiene una ponderación 0.1 y 2 tiene un 0.6, por lo que decimos que su acuerdo es 1 - |0.1 - 0.6| = 0.5. En X, 1 comparte 2 los clusters A, B y C, para los cuales tiene acuerdos 1 -

|0.6 - 0.4| = 0.8, 1 - |0.3 - 0.5| = 0.8 y 1 - |0.1 - 0.1| = 1, sumados 2.6. Entonces el *recall* para este par de instancias es *min(0.5, 2.6) / 0.5 = 1*. Continuando con el cálculo para todos los pares, sumándolos, y dividiendo por la cantidad de pares, se llega a que el *recall* para esta palabra es 1.
Para obtener la precisión se toma como referencia la clusterización en X, así para 1 se tienen los clusters A, B y C. Con la instancia 2 comparte en Y el cluster D, para el que ya se calculó el acuerdo, que es 0.5. En X, con la instancia 2 comparte los clusters A, B y C, para los cuales también se ha calculado un acuerdo de 2.6. Así que tenemos que la precisión para este par de instancias es *min(0.5, 2.6) / 2.6 = 0.19*. Si se calcula la precisión de esta manera para todos los pares de instancias, sumándolos, y dividiendo por la cantidad de pares, se llega a que la precisión para esta palabra es 0.28.
Para obtener la precisión y *recall* total, simplemente se promedia el de todas las palabras.

B.2. Fuzzy Normalized Mutual Information

La métrica *Fuzzy Normalized Mutual Information (FNMI)*, deriva de la información mutua definida formalmente como *I(X,Y)=H(X)-H(X|Y)* donde *H(X)* es la entropía de la variable aleatoria *X* que representa los conjuntos de instancias asignados a cada sentido. La información mutua típicamente se normaliza al intervalo [0,1] de forma de facilitar la comparación entre distintas clusterizaciones en una escala común, utilizando *Max(H(X),H(Y))* como factor normalizador.
Cada cluster X_i puede representarse como una variable aleatoria continua, con lo cual el *fuzzy cover* completo se denota como la variable $X_{1...k}$, donde la *i*-ésima entrada de *X* es la variable aleatoria continua para el cluster *i*.
Para evitar calcular la entropía en dominios continuos, puede computarse la información mutua mediante una discretización de los valores continuos de X_i . Como resultado, X_i pasa a ser una distribución categórica sobre un conjunto de rangos de aplicabilidad $\{w_1,...,w_n\}$ que denotan el grado de pertenencia en el conjunto de clusters.
Definiendo X_i y Y_j como variables categóricas sobre valores discretos, se puede estimar la entropía y la entropía conjunta como:

$$H(X_i) = \sum_{i=1}^{n} p(w_i) log_2 p(w_i)$$

Donde $p(w_i)$ es la probabilidad de una instancia de que su ponderación pertenezca al rango de aplicabilidad w_i. La entropía conjunta se define de forma similar como:

$$H(X_k, Y_l) = \sum_{i=1}^{n} \sum_{j=1}^{m} p(w_i, w_j) log_2 p(w_i, w_j)$$

Donde $p(w_i,w_j)$ es la probabilidad de una instancia de que su aplicabilidad pertenezca al rango w_i para el cluster X_k y w_j para el cluster Y_l , y *m* denota el número de categorías para Y_l. La entropía condicional entre dos clusters se calcula entonces como:

$$H(X_k|Y_l) = H(X_k, Y_l) - H(Y_l).$$

Estas dos últimas ecuaciones pueden usarse para definir *I(X,Y)* como en la definición original.

Volviendo al ejemplo de la fig. 19, tenemos el conjunto de etiquetas A, B, C y D en el *gold standard* y el conjunto A, B y C en el etiquetado producido por el sistema. Consideremos el vector de las ponderaciones de cada etiqueta en el *gold standard* para cada instancia, en el caso de A el vector es [0.7, 0, 0.4], para B es [0, 0.4, 0], para C es [0.2, 0, 0.6] y para D es [0.1, 0.6, 0]. Análogamente se tienen los vectores de ponderaciones para cada etiqueta en el etiquetado del sistema, en el caso de A es [0.6, 0.4, 0.7], para B es [0.3, 0.5, 0], y en el caso de C es [0.1, 0.1, 0.3]. Como primer paso, se calcula la entropía de cada etiqueta, tanto para la distribución en el etiquetado del sistema, como para la *gold standard*. Así, para el etiquetado del sistema, se tiene la entropía discretizada de su vector de ponderaciones calculada asignando cada entrada del vector en una partición del intervalo [0, 1] discretizado en 10 partes iguales. Luego para cada intervalo, se cuenta la cantidad de elementos pertenecientes, se divide por el total de elementos y se multiplica por su logaritmo en base 2. La sumatoria de estos valores es el opuesto de la entropía. En el caso de la etiqueta A en el etiquetado del sistema, tenemos un elemento en el intervalo (0.5, 0.6], un elemento en el intervalo (0.3, 0.4] y un elemento en el intervalo (0.6, 0.7], por lo que la entropía es $-1 \times \frac{1}{3} \times log_2(\frac{1}{3}) + \frac{1}{3} \times log_2(\frac{1}{3}) + \frac{1}{3} \times log_2(\frac{1}{3})$ =1.58. Análogamente para B la entropía discretizada es también 1.58, y para C es 0.92. Sumadas se tiene la entropía del conjunto de test, que es 4.09. Siguiendo idéntico procedimiento para el *gold standard* se obtiene que la entropía para este es 5.01.

Luego es necesario calcular la entropía condicional del *gold standard* dado el etiquetado del sistema. Tomando la etiqueta A en el *gold standard*, cuyo vector de ponderaciones es [0.7, 0, 0.4] conjuntamente con cada vector de ponderaciones en el etiquetado del sistema, se comparan entrada a entrada, y se considera que el par de entradas se correlaciona positivamente si ambos son 0 o ambos son mayores a 0, y negativamente si una de ellas es 0 y la otra no. Así comparado con el vector de ponderaciones de A en el etiquetado del sistema [0.6, 0.4, 0.7], la comparación entre las primeras entradas de ambos vectores es positiva, entre las segundas es negativa y entre las terceras es positiva. Así, la probabilidad de que una correlación sea positiva es $\frac{2}{3}$ y de que sea negativa es $\frac{1}{3}$, y las respectivas entropías son $-1 \times \frac{2}{3} log_2(\frac{2}{3}) = 0.39$ y $-1 \times \frac{1}{3} log_2(\frac{1}{3}) = 0.52$. Dado que la entropía de la probabilidad de que la correlación sea positiva es menor que la entropía de la probabilidad de que la correlación sea negativa, y que esto también ocurre al comparar el vector de ponderaciones en el *gold standard* para A con los vectores de ponderaciones en el etiquetado del sistema para B y C, el valor de la entropía condicional es igual al valor de la entropía discretizada del vector de ponderaciones de A en el *gold standard*, es decir 1.58. Lo mismo ocurre al comparar el vector de ponderaciones de C en el *gold standard* con los vectores del etiquetado del sistema, para el cual también la entropía es 1.58. En los casos de B y D, dado que existen casos donde la entropía de la probabilidad de la correlación negativa es menor que la de la probabilidad de una correlación positiva (en ambos casos en la comparación con el vector de B en el etiquetado del sistema) donde la entropía condicional es 0, calculada como el opuesto de la suma de las entropías conjuntas[22] menos

[22] Entropía calculada sobre los pares de intervalos en [0, 1] discretizado para los pares de entradas de los vectores de ponderaciones.

la entropía del vector del etiquetado del sistema. Por esto, la entropía condicional del *gold standard* respecto al etiquetado del sistema es 1.58+0+1.58+0=3.16.
Siguiendo un procedimiento análogo, la entropía condicional del etiquetado del sistema respecto al *gold standard* es 0.67+0.67+0.67=2.01.
Luego, el valor de *fuzzy normalized mutual information* es $\frac{4.09-3.16+5.01-2.01}{2} \times \frac{1}{max(4.09,\ 5.01)} = 0.39$