CONSIDERACIONES DE DISEÑO PARA UNA HERRAMIENTA DE ANÁLISIS SEMÁNTICO

DESIGN CONSIDERATIONS FOR A SEMANTIC ANALYSIS TOOL DEVELOPMENT

FRANCIS DE LA CARIDAD FERNÁNDEZ REYES

ffernandez@ceis.cujae.edu.cu

EXIQUIO C. LEYVA PÉREZ

exiquio@ceis.cujae.edu.cu

ROGELIO LAU FERNÁNDEZ

lau@ceis.cujae.edu.cu

Instituto Superior Politécnico “José Antonio Echeverría”. La Habana, Cuba.

RESUMEN

En el presente artículo se realiza un análisis de las múltiples clasificaciones que existen para los algoritmos de resolución de la ambigüedad en el sentido de las palabras, además se expone el recurso léxico fundamental utilizado para la implementación de éstos. A continuación se muestra el funcionamiento del algoritmo propuesto por Lesk en 1986, aplicado sobre un diccionario electrónico, así como su adecuación para el uso de la base de datos léxica *WordNet*. Luego se presenta el algoritmo de Lesk adaptado para hacer uso de la misma base de datos léxica y, finalmente, se procede a exponer las consideraciones de diseño que se tuvieron en cuenta en el momento de realizar la herramienta que incorpora los algoritmos antes mencionados.

Palabras clave: Lingüística computacional, análisis semántico, polisemia en las palabras.

ABSTRACT

The present paper accomplishes an analysis of the multiple classifications that exist for the word sense disambiguation algorithms (WSD); also it displays the lexical fundamental resource employed for the implementation of these. It shows the functioning of the algorithm proposed by Lesk in 1986, applied over an electronic dictionary, as well as his adjusting to the lexical data base *WordNet*. Later, it shows the adapted Lesk's algorithm using the same lexical data base and, finally, one proceeds to expose the designing

considerations that were taken into account to develop the software that incorporates above-mentioned algorithms.

Keywords: Computational linguistics, semantic analysis, word's polysemy.

Recibido: 10/01/2011. Aceptado: 16/03/2011

1. INTRODUCCIÓN

El procesamiento del lenguaje natural, como disciplina dentro de la lingüística computacional, involucra un conjunto de tareas y fases que evolucionan desde el análisis léxico de un texto hasta el análisis pragmático, donde se intenta descubrir las intenciones del autor. Un problema de los lenguajes naturales (español, inglés, entre otros) es la ambigüedad, tal y como se refleja en la siguiente oración: “Yo toco el bajo en los bajos de la escalera”. Este es un ejemplo clásico de ambigüedad, alguien que escucha esta frase entiende la intención del hablante, pero lo difícil es lograr que la computadora lo entienda. En la oración, anteriormente mostrada, se encuentra ambigüedad léxica, *bajos* está utilizado como adverbio de lugar, pero pudiera interpretarse que *bajos* sea sustantivo en plural y por tanto, la palabra *bajos* es un claro ejemplo de ambigüedad léxica. La resolución de la ambigüedad léxica forma parte del nivel de análisis léxico, a veces incluido en el análisis morfo-sintáctico de los textos. En el ejemplo anterior, también existe ambigüedad semántica, pues el sustantivo *bajo* puede tomar dos significados, indistintamente se puede ver como instrumento musical o como parte inferior de un pantalón.

En el caso de la ambigüedad léxica ya existen muchos trabajos que la tratan y la resuelven, casi en su totalidad, como parte del análisis léxico y sintáctico del procesamiento del lenguaje natural. Sin embargo un problema actual, aún sin resolverse completamente, es la ambigüedad semántica (Gelbukh y Sidorov, 2006; Navigli, 2009).

Por desambiguación del sentido de las palabras se entenderá la habilidad computacional de determinar cuál significado de una palabra debe ser activado por su utilización en un contexto en particular, es decir, se debe identificar el sentido correcto de palabras polisémicas en un texto dado (Navigli, 2009).

La desambiguación del sentido de las palabras (WSD) también puede ser vista como una tarea de clasificación: los sentidos de las palabras son clases, y un método de clasificación automática es usado para asignar a cada ocurrencia de la palabra una o más clases, basado en la evidencia del contexto y del uso de fuentes de conocimiento externas (Jurafsky & Martin, 2000).

Gelbukh y Sidorov (2006) definen la desambiguación del sentido de las palabras como el problema de escoger un sentido para una palabra usada en un texto,

además plantean que este proceso es muy popular en la lingüística computacional moderna.

En estas definiciones se apuntan dos elementos esenciales: considerar la desambiguación semántica como el proceso de asignación de un determinado sentido (predefinido o no) a una palabra; y que el proceso de determinar el sentido correcto depende del contexto donde se encuentre la palabra y de las fuentes de conocimiento externas a las que se tenga acceso.

Los métodos de desambiguación semántica pueden clasificarse de acuerdo al recurso principal empleado en este proceso (Mihalcea, 2006; Pedersen & Mihalcea, 2005). Los métodos basados en conocimiento son aquellos que utilizan, como recurso lingüístico principal para la desambiguación, a los diccionarios electrónicos, los tesauros, las ontologías y las bases de datos léxicas (Mihalcea, 2006).

Los métodos supervisados y semisupervisados son aquellos que utilizan como recurso lingüístico principal los corpus etiquetados con sentidos (Márquez, 2006). Estos métodos, en general, utilizan un conjunto de ejemplos que aparecen en el corpus para, a partir de ellos, generar las posibles clases –grupo de sentidos– de la desambiguación, con el objetivo de identificar patrones en los ejemplos asociados a cada clase en particular, generalizar estos patrones en reglas y aplicarlas ante el proceso de desambiguación de una nueva palabra, convirtiendo el proceso en una tarea de clasificación (Pedersen & Mihalcea, 2005). Generalmente su alcance reside en desambiguar una palabra por contexto, donde es necesario conocer la categoría gramatical de ésta, así como formularla explícitamente.

Los métodos no supervisados hacen uso de los corpus no etiquetados como recurso lingüístico principal (Pedersen, 2006). Estos métodos utilizan el aprendizaje no supervisado, que consiste en identificar patrones en grandes volúmenes de datos de ejemplo; primeramente se dividen los datos en grupos –clusters–, donde cada miembro de un grupo tiene más en común con los otros miembros de su propio grupo que con cualquier miembro de otro grupo diferente (Pedersen & Mihalcea, 2005).

En los métodos no supervisados no se dispone de sentidos de forma explícita y el proceso de desambiguación se reduce a diferenciar el uso de las palabras, discriminando entre los grupos y determinando cuáles instancias de los mismos tienen el mismo significado en el contexto, sin necesidad de etiquetar los sentidos de forma explícita, por lo que también se le conoce a este tipo de métodos como discriminación y constituyen una tarea de agrupamiento.

El Análisis Semántico Latente (LSA, por sus siglas en inglés) puede ser visto como un método de discriminación que encuentra similitudes semánticas entre los términos de determinados corpus de textos. Esta técnica está basada en procesos estadísticos que se establecen sobre un espacio semántico multidimensional, de esta forma las palabras de los textos son transformadas en un conjunto de vectores y luego se aplica una medida de similitud entre éstos, para determinar el grado

de relación semántica entre las palabras. Es por ello que este mecanismo ha sido utilizado como una forma de definir la coherencia entre términos de determinado contexto (Hernández Osuna y Ferreira Cabrera, 2010; Venegas V, 2003).

Finalmente, los métodos híbridos son aquellos que combinan características de más de un tipo de método específico.

Si se emplea la definición de los términos supervisados y no supervisados provenientes del aprendizaje automático (Nica, 2006), entonces un sistema de desambiguación semántica es supervisado si se debe entrenar sobre un conjunto de datos etiquetados con sentidos, y es no supervisado si hace uso de un conjunto de datos no etiquetados con sentidos. Esta es otra clasificación que aparece en la literatura consultada, sin embargo, los calificativos supervisado y no supervisado, en el contexto de la desambiguación semántica, no se emplean siempre de esta manera.

Se utiliza el término supervisado para hacer referencia a sistemas de desambiguación semántica basados en corpus etiquetados con sentidos; pero si las etiquetas son de distinta naturaleza, entonces se trata sólo de discriminación de sentidos (Schütze, 1998). Sin embargo, los enfoques no supervisados utilizan recursos lingüísticos secundarios –como la base de datos léxica *WordNet*– para ayudar en la obtención de los sentidos a partir de los datos no etiquetados. Estos métodos se pueden considerar entonces no supervisados, pues se basan en fuentes de conocimiento independientes, sin directa supervisión del fenómeno que se está aprendiendo. A pesar de cierta variabilidad en la bibliografía, generalmente se suele hablar de sistemas de desambiguación semántica supervisados si se entrenan con un corpus etiquetado con sentidos; en otro caso, si el corpus de entrenamiento tiene otro tipo de anotación o no tiene ninguna, entonces se habla de sistemas de desambiguación semántica no supervisados (Resnik & Yarowsky, 2000; Yarowsky, 2000).

Por otra parte, en la desambiguación semántica, la cuestión importante no es tanto la existencia o no de un conjunto de sentidos, sino la cantidad de datos de entrenamiento etiquetados requeridos por el algoritmo. La obtención de datos etiquetados con sentidos se considera actualmente el mayor obstáculo para la desambiguación semántica, de manera que la dependencia o no de estos datos se ha convertido en el criterio principal para la clasificación de estos sistemas, extendiéndose también a los métodos basados en conocimiento. En consecuencia, actualmente se tiende hacia la clasificación de los métodos de desambiguación semántica en sistemas supervisados, aquellos que necesitan ejemplos de entrenamiento etiquetados con sentidos para cada palabra a desambiguar, y sistemas no supervisados, aquellos que no necesitan tales ejemplos. A pesar de ciertas críticas, motivadas por el desconcierto que puede producir la redefinición de los términos de supervisado y no supervisado con respecto a su sentido originario en el aprendizaje automático (Stevenson, 2003), la nueva acepción para sistemas supervisados

y no supervisados parece imponerse (Berger, Della Pietra, & Della Pietra, 1996; Montoyo Guijarro, 2002; Pons Porrata, 2004; Resnik & Yarowsky, 2000; Rigau Claramunt, 2002; Yarowsky, 2000), por lo que será utilizada a partir de este momento en este trabajo.

El presente trabajo expone el diseño de una herramienta para la ejecución de algoritmos no supervisados en la desambiguación del sentido de las palabras y documenta los algoritmos seleccionados para la implementación. Siguiendo este propósito se divide en dos secciones, en la primera se expondrán los recursos y métodos de desambiguación semántica que se implementaron en la propuesta y posteriormente, en una segunda, se expondrán las consideraciones de diseño que se tuvieron en cuenta en el momento de la construcción de la aplicación informática que se propone para la experimentación con los algoritmos basados en recursos de conocimiento externos, principalmente en el uso de cualquier versión de *WordNet*. Además, finalmente se arriban a conclusiones y recomendaciones para la nueva versión de la aplicación informática.

2. RECURSOS Y ALGORITMOS UTILIZADOS EN LA HERRAMIENTA PROPUESTA

Los métodos no supervisados son aquellos que no utilizan ejemplos de entrenamiento etiquetados con sentidos, pueden utilizar recursos externos para adquirir conocimiento en cuanto a las definiciones de los conceptos que se intentan desambiguar; sin embargo los métodos que agrupan las palabras (con sentidos no predefinidos en fuente externa) en grupos y devuelven el grupo de palabras con el cual está relacionado cierto concepto objeto de la desambiguación se hacen llamar métodos discriminativos. Se han aplicado muchas hipótesis lingüísticas (heurísticas) en este tipo de métodos y se han utilizado múltiples recursos externos: *WordNet*, *eXtended WordNet*, *Brown Corpus*, entre otros. A continuación se exponen las características principales del recurso léxico *WordNet* y del algoritmo propuesto por Lesk en 1986 con diferentes adaptaciones realizadas a éste por parte de Banerjee, que sirvieron como base para la elaboración de la herramienta mencionada en este trabajo.

2.1. Base de datos léxica: *WordNet*

Las bases de datos léxicas son utilizadas en la Lingüística como fuentes de información a ser reutilizadas por otros recursos, por ejemplo, un lexicón computacional o una base de datos terminológica (Arano, 2003).

WordNet es una base de datos léxica diseñada sobre la base de las teorías psicolingüísticas del lexicón mental (Miller, Beckwith, Fellbaum, Gross, & Miller,

1990) con el objetivo de agilizar las búsquedas en los diccionarios en línea de la lengua inglesa, posteriormente, con el proyecto EuroWordNet (Vossen, 1996) se logró ampliarla para otros lenguajes, tales como: español, alemán, etc.

Esta base de datos léxica se construye sobre la base de las categorías sintácticas de sustantivo, verbo, adjetivo y adverbio y las relaciones semánticas de hiponimia, hiperonimia, meronimia, holonimia, sinonimia, antonimia, términos coordinados y troponimia. El equivalente a las relaciones de sinonimia y antonimia en el lenguaje natural se expresa mediante los sinónimos y antónimos de las palabras, respectivamente. Las relaciones de hiponimia e hiperonimia expresan relaciones “tipo de”, es decir, dados dos términos: *árbol* y *Pino*; *Pino* es un “tipo de” *árbol*, por lo que *Pino* es hipónimo de *árbol*, mientras que *árbol* es hiperónimo de *Pino*. Los términos coordinados se basan en la hiperonimia; se dice que dos términos son coordinados siempre que compartan un hiperónimo común, por ejemplo, *Pino* y *Ceiba* que comparten *árbol* como hiperónimo común. Las relaciones de meronimia y holonimia expresan relaciones “parte de”, es decir, dados dos términos: *carro* y *rueda*; *rueda* es “parte de” *carro*, por lo que *rueda* es merónimo de *carro*, mientras que *carro* es holónimo de *rueda*. La troponimia es una relación que se observa sobre verbos, se dice que dos verbos son tropónimos si uno de ellos se activa dentro del otro de alguna manera, por ejemplo, susurrar y hablar.

WordNet combina las características de varios recursos lingüísticos ya que incluye definiciones —o glosas— de términos en cada uno de sus sentidos, tal y como lo hace un diccionario y define conjuntos de palabras sinónimas, con diferentes relaciones semánticas entre ellas, tal y como lo hace un tesoro. Además, constituye un recurso de amplia utilización por parte de los algoritmos de desambiguación semántica (Pons Porrata, 2004) y su distribución se realiza de forma libre. WordNet se basa en el supuesto teórico de *matriz léxica*, integrada por los elementos *palabra* y *significado*. En la *matriz léxica* las columnas corresponden a palabras de un idioma y las filas a conceptos o significados. En la *matriz léxica*, los conceptos son representados por la lista de palabras que pueden ser usadas para expresarlo, es decir, por todos los elementos que pertenecen a una misma fila, los cuales constituyen sinónimos. Estas listas de palabras son llamadas “synsets” (término que proviene de conjunto de sinónimos).

2.2. Algoritmo de Lesk Original

Lesk (1986) creó una base de conocimiento que estaba asociada con cada sentido de una palabra en un diccionario y existía en ella una firma compuesta por la lista de palabras que aparecían en la definición del sentido en cuestión. La desambiguación tenía entonces como objetivo seleccionar el sentido de la palabra polisémica cuya firma contenga el mayor número de coincidencias con las firmas de las palabras vecinas, dentro de un contexto determinado (Ide & Véronis, 1998). El

algoritmo se basaba en dos suposiciones: primeramente, cuando dos palabras son usadas con una proximidad cercana en una oración, entonces ellas deben hacer referencia al mismo tópico, y segundo, cuando algún sentido de cada una de las dos palabras se puede utilizar para hacer referencia al mismo tópico, entonces sus definiciones deben compartir algunas palabras en común. Por ejemplo, cuando las palabras “*pine*” y “*cone*” ocurren juntas hacen referencias a “*evergreen*” y “*trees*” y, ciertamente, un significado de cada una de estas palabras contiene “*evergreen*” y “*tree*” en sus definiciones. Por tanto, se pueden desambiguar palabras vecinas en una oración comparando sus definiciones y escogiendo aquellos sentidos cuyas definiciones tengan el mayor número de palabras en común (Banerjee, 2002).

Formalmente (Navigli, 2009), dadas dos palabras en un contexto (w_1, w_2) , los sentidos $S_1 \in \text{Senses}(w_1)$ y $S_2 \in \text{Senses}(w_2)$ cuyas definiciones tengan el mayor solapamiento (más palabras en común, computado según la fórmula 1) son asumidos como los correctos:

$$\text{score}_{\text{Lesk}}(S_1, S_2) = |\text{gloss}(S_1) \cap \text{gloss}(S_2)| \quad (1)$$

Donde $\text{gloss}(S_i)$ es la bolsa de palabras de la definición textual del sentido S_i de la palabra w_i . Los sentidos que maximicen la fórmula anterior son asignados a cada palabra, respectivamente. Sin embargo, esto requiere el cálculo de $|\text{Senses}(w_1)| \times |\text{Senses}(w_2)|$ solapamiento de glosas. Si se extiende el algoritmo a un contexto de n palabras, se necesita calcular $\prod_{i=1}^n |\text{Senses}(w_i)|$, lo que resulta en una complejidad exponencial del algoritmo.

El método realizaba de forma correcta la desambiguación entre un 50-70% de las veces, utilizando un conjunto relativamente refinado de distinciones de sentidos, tales como aquellos que se encuentran en un diccionario de aprendizaje típico. El método de Lesk es muy sensible a la exactitud léxica en cada definición: la presencia o ausencia de una palabra puede radicalmente alterar los resultados. El algoritmo determina los solapamientos sólo entre las glosas de los sentidos que se están considerando, lo cual constituye una limitación, pues las definiciones de los diccionarios tienden a ser cortas y no proveen suficiente vocabulario para distinguir sentidos muy granulares. Sin embargo, el método de Lesk ha servido como base de la mayoría de los posteriores trabajos de desambiguación que hacen uso de diccionarios electrónicos (*Machine Readable Dictionaries, MRDs*) (Ide & Véronis, 1998).

2.3. Algoritmo de Lesk Extendido

Mientras que el algoritmo de Lesk restringe las comparaciones a las definiciones del diccionario de las palabras que están siendo desambiguadas, otra propuesta (Banerjee, 2002) selecciona como fuente de conocimiento a *WordNet*, lo que pre-

supone la comparación de las glosas de palabras que están conectadas por varias relaciones semánticas, con las palabras que están siendo desambiguadas. En los experimentos presentados por Banerjee (2002) se utilizaron las siguientes relaciones: hiperonimia, hiponimia, meronimia, holonimia, troponimia y atributo.

Entre las adaptaciones de Banerjee se encuentra la selección de las glosas a comparar, pues existen varias glosas que surgen por la utilización de las relaciones semánticas de *WordNet*. Para ello se plantean dos esquemas de selección que se nombran Heterogéneo y Homogéneo. El primero se basa en la comparación de cada glosa (producto de múltiples relaciones semánticas) asociada con un sentido en particular de la primera palabra con cada glosa asociada con algún sentido de la segunda. En el primer esquema la cantidad de comparaciones depende de la cantidad de relaciones encontradas para un sentido en particular. Siendo w_1, w_2 dos palabras objeto de desambiguación, $S_1 \in Senses(w_1)$ y $S_2 \in Senses(w_2)$ dos sentidos de las palabras anteriores, n el número de glosas extraídas de S_1 mediante múltiples relaciones semánticas y m definida de forma similar para S_2 , entonces la cantidad de comparaciones viene dada por $n \times m$.

Por otro lado, el esquema Homogéneo es una extensión de los experimentos de Lesk, donde se comparan dos sentidos de dos palabras objeto de desambiguación, se buscan solapamientos entre sus glosas y también entre las glosas que son extraídas por la aplicación de una misma relación a ambos sentidos objeto de análisis. Para determinar la cantidad de comparaciones se hace uso de las mismas definiciones anteriores y es expresada por el $\min(n, m)$.

Cuando se comparan las definiciones del diccionario de dos palabras, el algoritmo de Lesk original asigna una medida igual al número de palabras ("tokens") que ocurren en ambas glosas. Banerjee (2002) introduce un mecanismo nuevo para establecer la medida de relación que se establece entre las glosas, basado en el solapamiento, pues si ocurre un solapamiento de n palabras consecutivas entre dos glosas, ese fenómeno debe recibir una medida mayor que si ocurren de forma separada en las glosas. El nuevo mecanismo apela a la ley de Zipf (Adamic & Huberman, 2002) que establece que la frecuencia de un evento es inversamente proporcional al orden de aparición del evento, donde el orden es calculado basado en la frecuencia de todos los eventos. Esto implica que la mayoría de los eventos ocurren sólo una vez, y sólo unos pocos ocurren con mayor frecuencia. Por tanto, si se encuentra un solapamiento de más de una palabra de forma consecutiva, éste es un evento notable y merece más que un incremento lineal en la medida de relación entre las glosas. Es por ello que Banerjee define esta medida con un incremento cuadrático del número de palabras que ocurren en un solapamiento.

El algoritmo de Lesk original desambigua cada término por separado: para cada palabra de la ventana contextual, compara la glosa de cada sentido de esta palabra con la glosa de todos los sentidos de todas las palabras alrededor de ella y le asigna el sentido que produce el mayor solapamiento; luego reinicia el pro-

ceso de nuevo con la próxima palabra de la ventana contextual. Por lo tanto, el sentido asignado a una palabra del contexto no depende de conocer a priori los sentidos particulares de sus vecinas, a este enfoque se le llama enfoque Local, pues la desambiguación de varias palabras en una oración es tratada como una serie de problemas independientes que no tienen efecto uno en el otro. Banerjee (2002) propone un enfoque donde todas las palabras de la ventana contextual son desambiguadas simultáneamente con el objetivo de obtener la mejor combinación de sentidos para todas las palabras de la ventana, en vez de sólo para la palabra objeto de la desambiguación. Contrariamente al enfoque Local, el enfoque Global elige el sentido de la palabra objeto de la desambiguación teniendo en cuenta aquellos seleccionados para las palabras vecinas.

Se demostró (Banerjee, 2002) que el enfoque Global funciona correctamente cuando las palabras en la ventana contextual se encuentran estrechamente vinculadas, y existen oportunidades de encontrar relaciones semánticas entre las palabras no objeto del proceso de desambiguación. Sin embargo, en oraciones largas donde las palabras no están fuertemente vinculadas es mejor utilizar el enfoque Local que obvia las comparaciones entre sentidos de palabras no objeto de la desambiguación.

3. CONSIDERACIONES DE DISEÑO

La herramienta que se propone se instrumentó en lenguaje Java, con el Entorno de Desarrollo Integrado (“Integrated Development Environment”, IDE) NetBeans 6.0, por lo cual es fácilmente portable a otras plataformas. Su desarrollo y las pruebas a ésta se desplegaron sobre el sistema operativo Windows.

En la aplicación se adaptaron dos recursos lingüísticos fundamentales para su correcto funcionamiento, permitiendo la experimentación con los algoritmos sobre dos idiomas, principalmente: inglés y español. Para el idioma inglés se cargan los ficheros de datos de *WordNet* (versión 1.5 o superior), lo cual es configurable por el usuario y se implementó un algoritmo de búsqueda binaria para obtener alguna información de estos ficheros, disminuyendo así el tiempo de búsqueda. Sin embargo, para el español se accede a *WordNet* 1.6 a través de una base de datos en Postgres, pero como carece de muchas glosas, se decidió no realizar experimentos debido a que las variantes de Lesk se basan fundamentalmente en las definiciones de los términos. Para que el usuario configure el idioma con el que desea experimentar se tiene un fichero “config.properties” que es editable y es responsable de manipular las propiedades del sistema. Se utiliza, además, el patrón de diseño “Factory Method” para adaptar nuevos recursos a la herramienta. A continuación se muestra el diagrama de diseño de las clases encargadas de manipular los recursos de *WordNet* en inglés:

Figura 1. Diagrama de clases de diseño del recurso de *WordNet* en inglés.

En la Figura 1 se observa el enumerado “TypePosFile”, encargado de definir las distintas categorías gramaticales existentes en la base de datos léxica “WordNet”: adjetivos, adverbios, sustantivos y verbos. A su vez, las clases “DataFileFormat” e “IndexFileFormat” son las encargadas de cargar los ficheros provenientes de la base de datos léxica en memoria interna para que los algoritmos implementados hagan uso de éstos.

Los algoritmos que se implementaron son los siguientes: Lesk Original y la adaptación propuesta por Banerjee que encierra dos enfoques, el global y el local. Todos los algoritmos implementan una interfaz que contiene un método encargado de la definición del proceso de desambiguación, donde cada algoritmo lo implementa según sus características.

Los patrones de diseño son la base de la búsqueda de soluciones a problemas comunes del desarrollo de software, particularmente los que emergen producto del diseño de los módulos del sistema informático que se desee implementar. Estos patrones se clasifican de acuerdo al objetivo que persiguen en: estructurales, crea-

cionales o de comportamiento y el grupo “Gang of Four” (GoF, por sus siglas en inglés) fue quien los propuso en la década del 90. La herramienta propuesta utiliza varios patrones de diseño de tipo creación, entre ellos es conveniente destacar el “Factory Method” y el “Singleton”. El primero se encarga de desarrollar métodos que permitan crear instancias de clases en tiempo de ejecución, mientras que el segundo asegura que exista una única instancia de la clase que lo implementa. El patrón de diseño “Factory Method” se utilizó para adaptar nuevos algoritmos a la herramienta, permitiendo la flexibilidad y el patrón de diseño “Singleton” para crear un único punto de acceso a la aplicación, evitando así la duplicidad de información y el malgasto de memoria interna. A continuación se muestra el diseño de las clases encargadas de manipular los algoritmos implementados:

Figura 2. Diagrama de clases de diseño de los algoritmos no supervisados: paquetes que modelan el comportamiento individual de cada algoritmo implementado.

En la Figura 2 se aprecia el enumerado *TypeAlgorithm* que define los algoritmos que se encuentran implementados en la herramienta propuesta, entre éstos se encuentran las versiones del algoritmo de Lesk (englobados con el tipo *ConcreteAdaptedLeskCreator* e implementados en el paquete *lesk*). Igualmente, se enriqueció la herramienta con otros algoritmos no supervisados: basado en los dominios de Magnini (el tipo *WordNetDomains* de *TypeAlgorithm* indica que estas variantes son codificadas dentro del paquete *wn-domains*). La clase *GRIALCreator* define un método combinado basado en votos que utiliza las variantes del algoritmo de Lesk, los dominios de Magnini y una función de pesado sobre las diferentes relaciones semánticas que aparecen en *WordNet*.

El paquete *baseline*, que aparece representado en la Figura 2, agrupa un diseño de clases para la ejecución de heurísticas de referencias que permiten evaluar los resultados del proceso de desambiguación semántica de los algoritmos no supervisados. La clase *ConcreteAlgorithm* es la encargada de crear el algoritmo que se desea ejecutar por medio del enumerado *TypeAlgorithm* que se corresponde con el nombre de la clase que implementa el algoritmo en particular. Este es un ejemplo claro de la flexibilidad de la aplicación informática, donde para incorporar un nuevo algoritmo sólo se debe agregar el nuevo valor al enumerado que se corresponda con el nombre de la clase que implementa el algoritmo en cuestión y esta incorporación sería responsabilidad del desarrollador. En la página siguiente se muestra el diagrama de clases de diseño del paquete *lesk* que apareció en la Figura 2.

La Figura 3 muestra el enumerado *TypeDisambiguationStrategy* que contiene las dos estrategias del algoritmo de Lesk Extendido: Global o Local que son implementadas por las clases *LocalStrategy* y *GlobalStrategy*, respectivamente. Además, la clase *ConcreteAdaptedLeskCreator* es la encargada de ejecutar las dos variantes del algoritmo de Lesk: Extendido y Original. Por otra parte, la clase *ConcreteAdaptedLeskFactory* utiliza el patrón de diseño “Factory Method” para crear una de las estrategias contenidas en *TypeDisambiguationStrategy*. Las clases *BanerjeeAdaptedLeskMechanism* y *OriginalLeskMechanism* son las encargadas de efectuar el puntaje de acuerdo al solapamiento de las glosas. La primera utiliza la Ley de Zipf para efectuar la votación, mientras que la segunda cuenta la cantidad de intersecciones que se producen entre las glosas, tal como hace el algoritmo de Lesk original.

Las palabras tradicionalmente se han agrupado en clases de equivalencia, llamadas partes del discurso, partes de la oración o categorías gramaticales. Esta clasificación (Muñoz Porras, 2008) se basa en las funciones morfológica y sintáctica de la palabra, es decir, las clases de palabras se reconocen tanto por su función en la oración como por sus características morfológicas. La morfología se ocupa de la estructura interna de la palabra. La sintaxis se encarga de regular la combinación de las palabras dentro de la oración.

Figura 3. Diagrama de clases de diseño de las variantes de implementación del algoritmo de Lesk.

Los etiquetadores morfosintácticos, tal como establece Liddy (2001), tratan con las componentes de las palabras, las cuales se identifican como morfemas –las unidades más pequeñas de significado. A los morfemas se les puede agregar prefijos y sufijos, por tanto, un etiquetador morfosintáctico es aquel que reconoce las oraciones, reconoce las palabras dentro de las oraciones y para cada palabra es capaz de extraer los morfemas, separando los prefijos y sufijos del mismo, además establece para cada palabra la categoría gramatical correcta de acuerdo a su uso en el texto y otro conjunto de marcas léxicas como son: modo, tiempo, persona para los verbos, entre otras. Los etiquetadores morfosintácticos pueden ser de dos tipos: basados en reglas (modo experto) o basados en métodos estocásticos (modo empírico).

Como parte del preprocesado de los textos, se hizo necesaria la adecuación de algún etiquetador morfosintáctico. Con el objetivo de reutilizar analizadores existentes se creó una nueva interfaz *POSTagger* capaz de definir el proceso de extracción de formas raíces y asignación de categorías gramaticales a las palabras necesario para utilizar los algoritmos de desambiguación semántica implementados. Haciendo uso de esta interfaz de software se acoplaron los etiquetadores FreeLing, dependiente de la plataforma Windows y el analizador MorphAdorner, que es implementado en lenguaje Java y por tanto multiplataforma.

El paquete FreeLing (Padro, 2007) consiste en una biblioteca que provee servicios para el análisis del lenguaje (tales como análisis morfológico, reconocimiento de fechas, etiquetador sintáctico, etc.) La versión 2.0 del paquete provee segmentación, división de oraciones, análisis morfológico, clasificación y detección de entidades nombradas, reconocimiento de fechas/ números/ magnitudes físicas, etiquetador sintáctico, análisis sintáctico de superficie y anotación de sentidos obtenidos de WordNet. FreeLing está diseñado para ser utilizado como librería externa en cualquier aplicación que necesite este tipo de servicios. La distribución incluye diccionarios en diferentes idiomas (inglés, español, catalán, gallego e italiano). FreeLing permite realizar el análisis morfosintáctico empleando alguno de los siguientes algoritmos: Modelos Ocultos de Markov (Brants, 2000) (“Hidden Markov Model”, HMM) o el etiquetado Relajado (Padro, 1997) (“Relaxation Labeling”, RL). Esta aplicación emplea un conjunto de etiquetas para representar la información morfológica de las palabras. Este conjunto de etiquetas se basa en las etiquetas propuestas por el grupo EAGLES¹ para la anotación morfosintáctica de lexicones y corpus para todas las lenguas europeas.

MorphAdorner (University, 2009) es un programa de línea de comando en Java que actúa como gestor de tuberías para los procesos que desarrollan las anotaciones morfológicas en un texto. Un gestor de tuberías es un programa que ejecuta múltiples procesos simultáneamente, de forma planificada y evitando colisiones entre estos debido al acceso de un recurso común. Actualmente este programa provee métodos para anotar textos con categorías gramaticales y formas raíz. También provee facilidades para segmentar textos, reconocer los límites de las oraciones y extraer nombres y lugares. El conjunto de etiquetas que utiliza este etiquetador son las categorías gramaticales de NUPOS² diseñadas por Martin Mueller, que funcionan para el griego y el inglés. MorphAdorner provee muchos algoritmos para hacer el etiquetado morfosintáctico, pero sólo dos son ampliamente usados y documentados:

- El etiquetador de trigramas que utiliza un HMM y una variante de búsqueda

¹ <http://www.ilc.cnr.it/EAGLES96/home.html>.

² <http://panini.northwestern.edu/mmueller/nupos.pdf>.

queda en haz sobre el algoritmo de Viterbi, se presupone sea el analizador primario.

- El etiquetador basado en reglas, que es una versión modificada del etiquetador de Mark Hepple. Este etiquetador, a su vez, es una variante del etiquetador de Eric Brill, pero deshabilita la interacción entre las reglas. Este etiquetador es utilizado como secundario para corregir la salida del etiquetador basado en trigramas.

El algoritmo de Lesk original se implementó para ventanas de contexto variables, sustituyendo las definiciones que se obtenían del diccionario por la glosa que se obtiene de WordNet. Sin tener en cuenta los ejemplos de utilización de la palabra que también provee WordNet, se realiza un proceso de extracción de formas raíces de cada palabra de las glosas para minimizar los errores provenientes de la exactitud léxica cuando se procede a comparar las glosas, y se devuelve el sentido que más palabras comparte con los sentidos de las palabras del contexto, Si no se comparten palabras en común, entonces no se devuelve ningún sentido, pues existe ausencia de evidencias para devolver alguno de ellos en particular.

Según los resultados experimentales de Banerjee (2002), y sus conclusiones sobre la precisión de los esquemas, sólo se implementó en la herramienta el Heterogéneo, aunque es más costoso computacionalmente.

4. CONCLUSIONES

La explosión combinatoria del algoritmo original de Lesk provoca costos de memoria y tiempo en la obtención del sentido correcto de una determinada palabra y por ello han surgido nuevas variantes de algoritmos que intentan solucionar este problema. Entre estas variantes está el algoritmo de Lesk Extendido que emplea glosas enriquecidas por determinadas relaciones semánticas extraídas de WordNet y aplica la Ley de Zipf para calcular el solapamiento de éstas, pero no utiliza con proporciones adecuadas las relaciones existentes entre los términos.

La biblioteca de clases, presentada en el presente trabajo, se caracteriza por la flexibilidad que ofrece para la incorporación de nuevos algoritmos y recursos léxicos. Lo antes señalado se logra mediante la utilización del patrón de diseño “Factory Method” y el uso de las clases del paquete “reflection” de Java. El diseño de esta herramienta ofrece bajo acoplamiento entre sus módulos, lo cual reporta un beneficio adicional al reconfigurar los algoritmos no supervisados sin necesidad de re-codificarlos.

5. RECOMENDACIONES

Se propone, para mejorar el rendimiento de la herramienta propuesta, la aplicación de diferentes técnicas de implementación, entre ellas se pueden citar: la programación multihilos, la utilización de tablas hash con llaves conformadas por synsets para evitar el preprocesado múltiples veces de una determinada glosa y la utilización de librerías para que el tratamiento de cadenas de texto sea más eficiente, debido a que los principales problemas detectados en la herramienta se encuentran en el tratamiento de cadenas de texto, en el preprocesado de las glosas y en la explosión combinatoria de sentidos que se manifiesta cuando la ventana contextual aumenta.

La programación multihilos permite la ejecución de varias tareas simultáneamente, lo que mejora el rendimiento del sistema expresado por el aumento en la velocidad de respuesta de la herramienta propuesta. Las tablas hash, por otra parte, también mejoran la velocidad de respuesta ya que constituyen una estructura de datos que asocia una llave (dato único) con uno o varios valores, de esta forma las búsquedas e inserciones de valores se realizan más rápidamente.

REFERENCIAS

- Adamic, L. A. & Huberman, B. A. 2002. Zipf's law and the Internet. *Glottometrics* 3, pp. 143-150.
- Arano, S. 2003. "La ontología: una zona de interacción entre la Lingüística y la Documentación", en *Hipertext.net*, 2. Disponible en: <http://www.hipertext.net/web/pag220.htm>. Acceso: 15/05/2008.
- Banerjee, S. 2002. *Adapting the lesk algorithm for word sense disambiguation using wordnet*. Minnesota, USA: University of Minnesota.
- Berger, A. L., Della Pietra, S. A. & Della Pietra, V. J. 1996. A Maximum Entropy Approach to Natural Language Processing. *Computational Linguistics* 22 (1), pp. 1-36.
- Brants, T. 2000. TnT -- A Statistical Part-of-Speech Tagger. Paper presented at the Proceedings of the Sixth Conference on Applied Natural Language Processing.
- Gelbukh, A. y Sidorov, G. 2006. *Procesamiento automático del español con enfoque en recursos léxicos grandes*. México: Centro de Investigación en computación, Instituto Politécnico Nacional.
- Hernández Osuna, S. y Ferreira Cabrera, A. 2010. "Evaluación automática de coherencia textual en noticias policiales utilizando análisis semántico latente", en *RLA: Revista de Lingüística Teórica y Aplicada* 4 (2), pp. 115-139.
- Ide, N. & Véronis, J. 1998. "Word Sense Disambiguation: The State of the Art", en *Computational Linguistics*, 24 (1), pp. 1-41.
- Jurafsky, D. & Martin, J. H. 2000. *Speech and Language Processing: An introduc-*

- tion to natural language processing, computational linguistics, and speech recognition*. Second edition. New Jersey: Prentice Hall.
- Lesk, M. 1986. "Automatic sense disambiguation using machine readable dictionaries: How to tell a pine cone from an ice cream cone". Paper presented at the 5th annual international conference on Systems documentation, Ontario, Canadá.
- Liddy, E. D. (Ed.). 2001. *Encyclopedia of Library and Information Science*. Second edition. New York: Marcel Decker, Inc.
- Márquez, L. 2006. Supervised Corpus-Based Method for WSD. In E. Agirre & P. Edmonds (Eds.), *Word Sense Disambiguation, Algorithms and Applications*. Springer, pp. 167-216.
- Mihalcea, R. 2006. "Knowledge-Based Methods for WSD". En E. Agirre & P. Edmonds (Eds.), *Word Sense Disambiguation, Algorithms and Applications* Springer. pp. 127-132.
- Miller, G. A., Beckwith, R., Fellbaum, C., Gross, D. & Miller, K. 1990. "Introduction to WordNet: An on-line lexical database", en *International Journal of Lexicography* 3, pp. 235-244.
- Montoyo Guijarro, J. A. 2002. Desambiguación léxica mediante Marcas de Especificidad. Tesis de Doctorado, Universidad de Alicante, España.
- Muñoz Porras, V. 2008. *Herramientas para la Extracción de Información bajo la arquitectura GATE*. México: Universidad de Guanajuato.
- Navigli, R. 2009. "Word sense disambiguation: A survey", en *ACM Computing Surveys* 41 (2), pp. 1-69.
- Nica, I. 2006. El conocimiento lingüístico en la desambiguación semántica automática. Tesis de Doctorado. Universidad de Barcelona, España.
- Padro, L. 1997. *A Hybrid Environment for Syntax-Semantic Tagging*. Barcelona: Universidad Politécnica de Cataluña.
- Padro, L. 2007. *FreeLing User Manual*. Barcelona: Universidad Politécnica de Cataluña.
- Pedersen, T. 2006. Unsupervised Corpus-Based Methods for WSD. In P. E. Eneko Agirre (Ed.), *Word Sense Disambiguation, Algorithms and Applications*. Springer, pp. 133-166.
- Pedersen, T., & Mihalcea, R. 2005. Advances in Word Sense Disambiguation. Paper presented at the Association for Computational Linguistics (ACL'05).
- Pons Porrata, A. 2004. Desarrollo de algoritmos para la estructuración dinámica de información y su aplicación a la detección de sucesos. Tesis de Doctorado. Universidad Jaume I, Castellón, España.
- Resnik, P. & Yarowsky, D. 2000. "Distinguishing Systems and Distinguishing Senses: New Evaluation Methods for Word Sense Disambiguation", en *Natural Language Engineering* 5 (3), pp. 113-133.
- Rigau Claramunt, G. 2004. "Resolución automática de la ambigüedad semántica

- de las palabras". En M.A.M Antonín (Ed.), *Tecnologías del texto y del habla*. Universitat Barcelona Ed. Vol. 72, pp. 253.
- Schütze, H. 1998. "Automatic Word Sense Discrimination", en *Computational Linguistics* 4 (1), pp. 97-124.
- Stevenson, M. 2003. *Word Sense Disambiguation: The case for combinations of Knowledge Sources*. Standford: CSLI Publications.
- University, N. 2009. MorphAdorner: A Java Library for the Morphological Adornment of English Language Texts.
- Venegas V, R. 2003. "Análisis Semántico Latente: una panorámica de su desarrollo", en *Revista Signos* 36, pp. 121-138.
- Vossen, P. 1996. "EuroWordNet: Building a Multilingual WordNet Database with Semantic Relations between Words", en *Procesamiento del lenguaje natural* 18, pp. 145-158.
- Yarowsky, D. 2000. "Hierarchical Decision Lists for Word Sense Disambiguation", en *Computers and the Humanities* 34, pp. 179-186.