Desambiguación de sentidos de palabras usando sinónimos

Trabajo terminal No. |2007-0016|
Alumno: Ríos Gaona Miguel Ángel

fortimbras_82@hotmail.com

Directores: Dr. Alexander Gelbukh, M. en C. Macario Hernández Cruz

Resumen. La desambiguación de sentidos de las palabras se define como el problema computacional de, dados un texto y (una ocurrencia de) una palabra en este texto, determinar automáticamente cual de sus acepciones (las llamamos sentidos), entre las listadas en un diccionario dado, es la que el autor tenía en mente. En este trabajo se proponen dos métodos no supervisados. El primero usa palabras (cuasi)sinónimas de cada sentido, generalizando el método de Mihalchea y Moldovan (1999). El segundo es una modificación del algoritmo de Lesk (1986) que usa la tasa de coocurrencias en el Internet. Se demuestra la utilidad del primer método a través de su aplicación en un sistema de traducción automática de tipo palabra por palabra.

Índice

1. Introducción

La ciencia que estudia el lenguaje humano se llama lingüística. En esta ciencia existen ramas que representan su intersección con otros campos, del conocimiento científico como de la tecnología, la educación, la medicina, el arte y otras actividades humanas.

En particular, existe una relación muy especial e interesante –de gran beneficio mutuo– entre la lingüística y la computación.

Pro un lado, el conocimiento lingüístico es la base teórica para el desarrollo de una amplia gama de aplicaciones tecnológicas, de cada vez mas alta importancia.

Por otro lado, las tecnologías computacionales pueden dotar a la lingüística de herramientas con las que no contaban los investigadores de las dos décadas anteriores.

El amplio campo de intersección e interacción entre la lingüística y la computación se estructura a su vez en varias ciencias más específicas. Una de ellas es la lingüística computacional. Esta ciencia trata de la construcción de modelos de lenguaje entendibles para las computadoras, es decir, más formales que los modelos tradicionales orientados a lectores humanos. Otra área es el procesamiento automático del lenguaje natural (PNL), que se ocupa más de los aspectos técnicos y algoritmos de la aplicación de dichos modelos a los grandes volúmenes de texto, con el fin de estructurarlos según la información contenida en ellos, de extraerles información útil y de transformar esta información. Estas dos disciplinas tienen el mismo objeto de investigación, aunque lo consideran desde enfoques diferentes.

1.1. Ubicación

¿Qué más importante saber en lingüística para desarrollar modelos que sean aptos para las computadoras? Se puede tratar de desarrollar un modelo de lenguaje completo, sin embargo, es preferible dividir el objeto en partes y construir modelos más pequeños y por ello más simples, de partes del lenguaje. Para eso se usa el concepto de niveles del lenguaje. Tradicionalmente, el lenguaje se divide en seis niveles:

Fonética / Fonología. Es la parte de la lingüística que se dedica a la exploración de las características del sonido, que es un elemento substancial del lenguaje. Eso determina que los métodos de fonética sean en mayoría físicos; por eso su posición dentro del la lingüística es bastante independiente.

Morfología. El área de morfología es la estructura interna de las palabras (sufijos, prefijos, raíces, flexiones) y el sistema de categoría gramaticales (género, número, etc.).

Sintaxis. Se dedica a analizar las relaciones entre las palabras dentro de la frase.

Semántica. El propósito de la semántica es "entender" la frase. ¿Pero qué significa "entender"? Hay que saber el sentido de todas las palabras e interpretar las relaciones sintácticas. Una aplicación importante del análisis semántico es la desambiguación de sentidos de palabras (WSD: *Word Sense Disambiguation*). Siendo el análisis semántico el objetivo de estudio de este trabajo terminal.

Pragmática. Usualmente se dice que la pragmática trata de las relaciones entre la oración y el mundo externo.

Discurso. Normalmente no hablamos con una oración aislada, sino con varias oraciones. Esas oraciones tienen ciertas relaciones entre si. Las oraciones hiladas forman una nueva entidad llamada discurso.

1.2. Definición del problema

Las palabras comúnmente tienen múltiples sentidos. Los seres humanos generalmente son buenos para seleccionar cual sentido de una palabra es el deseado en un contexto en particular.
Por ejemplo.

Juan dejó el periódico en el banco.
Se sentó en el banco.
Entro en el banco y fue a la ventanilla.

En muchas ocasiones es muy útil si el software pudiese distinguir entre diferentes sentidos de una palabra. En el campo de la lingüística computacional, este problema es generalmente llamado desambiguación de sentidos de palabras, y es definido como el problema computacional de determinar cual "sentido" de una palabra es activado por el uso de esa palabra en un contexto particular.

La WSD es en esencia una tarea de clasificación: Los sentidos de la palabra son las clases, el contexto provee la evidencia y cada ocurrencia de una palabra es asignada a una o más de las posibles clases basado en la evidencia.

1.3. Justificación y aplicaciones

La importancia de la WSD ha sido reconocida en la computación lingüística; con al menos 700 artículos en la antología de la ACL (*Association for Computational Linguistics*).

Pero la tarea no es pensada como un fin en sí misma sino como un mejorador para otras tareas y aplicaciones de la computación lingüística y del procesamiento del lenguaje natural

Traducción Automática (MT: *Machine Translation*). La desambiguación es requerida para palabras que tienen diferentes traducciones para diferentes sentidos que son potencialmente ambiguos en un dominio determinado. Por ejemplo, traducir *bill* del inglés al español. ¿Es un pico o una cuenta?

Recuperación de información (IR: *Information retrieval*) La ambigüedad tiene que ser resuelta en algunas consultas. Por ejemplo para la consulta *depresión* el sistema debe retornar documento acerca de enfermedad o economía. Un problema similar proviene de los sustantivos propios como *Raleigh* (bicicleta, ciudad, persona, etc.).

Los sistemas actuales de IR no utilizan explícitamente la WSD, y la tarea recae en que el usuario de suficientes palabras para el contexto de la consulta, para recuperar documentos relevantes en el sentido deseado *depresión tropical*.

1.4. Aportaciones

Este trabajo terminal propone e implementa dos métodos no supervisados para la solución de la WSD y una aplicación de uno de estos métodos. A saber, las aportaciones principales son:

– Un **método no supervisado para la WSD**, que generaliza el método de Mihalchea y Moldovan (1999). El método se basa en el uso de la Web y de un diccionario de las palabras semánticamente cercanas a la palabra ambigua.

– Un **método no supervisado para la WSD**, que modifica el algoritmo de Lesk (1986). El método se basa en el uso de las estadísticas de las coocurrencias de las palabras. Estas estadísticas se colectan a través de la Web. El método modificado proporciona mejor precisión (*accuracy*) que el método original.

– Aplicación del primero de estos métodos, a través del diseño e implementado de una aplicación del PLN, en especifico una **aplicación de traducción automática** palabra por palabra.

1.5. Estructura del documento

CAPÍTULO 2 El reporte comienza con una revisión de los métodos ya existentes para la solución de la tarea de WSD.

En el siguiente capitulo se muestra el marco teórico que servirá para la formulación del problema.

En el capitulo 4 se proponen los nuevos métodos, pasando después al diseño implementación del primer método en una aplicación del PLN.

En el siguiente capitulo se muestran los resultados experimentales así como su comparación contra otros métodos.

Finalizando con las conclusiones, trabajo futuro y la publicación preparara.

2. Antecedentes

El problema de la WSD se encuentra dentro de la área de la inteligencia artificial (IA), y en concreto, se considera como un problema de IA-completo (*AI-complete*) lo que presupone para su solución un completo entendimiento del lenguaje.

Así, los primeros enfoques de la WSD en los años setenta y ochenta proceden de esta área del conocimiento. En este período, el conocimiento necesario para la WSD se codificaba a mano y se limitaba a conjuntos restringidos de palabras. Los modelos que se usaban eran desarrollados a propósito para la tarea. Especialmente los años noventa están marcados por la aparición de lexicones computacionales (diccionarios en formato electrónico) la gran cantidad de datos almacenados permiten la utilización de técnicas de tipo estadístico.

Por otra parte se explotan las fuentes del conocimiento y empiezan a ser más utilizados los métodos basados en corpus, sobre todo a partir del momento en que se dispone de corpus etiquetados semánticamente (SemCor, DSO, Line, etc.). Los corpus anotados permiten la derivación de conocimiento necesario para la resolución del problema y para este propósito se aplican técnicas de aprendizaje automático (ML: *Machine Learning*).

A partir de los años noventa, se investigan diferentes tipos de algoritmos y fuentes de información, con el objetivo de mejorar el proceso de desambiguación.

En resumen, en el área de WSD se encuentran enfoques desde áreas tan diversas como la IA (modelos simbólicos, como árboles de decisión y subsimbólicos como redes neuronales), modelos estadísticos (clasificadores bayesianos, cadenas de Markov, Máxima entropía) técnicas de ML (aprendizaje basado en ejemplos y técnicas de teoría del aprendizaje computacional).

2.1. Información usada

El problema de los recursos léxicos es fundamental para la investigación en el área así como para elegir que métodos utilizar para su solución.

Fuentes de conocimiento léxico. Entre las fuentes más utilizadas se encuentran las estructuradas, las no estructuradas y las combinaciones de ambas. A la primera clase pertenecen obras elaboradas fuera del ámbito de la WSD, de uso general que suelen tener una estructura. La segunda corresponde a los corpus, estas no tienen una organización en términos de palabras o sentidos.

Las fuentes léxicas estructuradas se pueden clasificar en los siguientes tipos:

- Diccionarios
- Tesauros
- Ontologías o redes semánticas
- Lexicones generativos

Los diccionarios accesibles por computadora (MRD, *Machine Readable Dictionaries*) aparecen en los años ochenta. Representan la fuente más estudiada y utilizada en la investigación en la WSD, y en general para la adquisición de conocimiento necesario para las tareas del PLN. La unidad básica es el sentido, al cual le corresponden el campo de la definición y otros campos opcionales, como etimología, sinónimos, ejemplos, etc.

Los MRD tienen como mayores desventajas la inconsistencia de las definiciones, la granularidad inadecuada para la WSD y la falta de información para la determinación del sentido.

Un tesauro es esencialmente una organización jerárquica de listas de palabras principalmente sinónimo. Suele estar organizados alfabéticamente y se distingue de los diccionarios en tanto que no define los sentidos sino que se limita a proveer sus sinónimos y relaciones de hiponimia e hiperonímia (relación clase-subclase). El tesauro más utilizado en la WSD es el *Roget´s International Thesaurus*.

Las redes semánticas constituyen representaciones del conocimiento estructurado, organizadas como grafos con enlaces etiquetados entre nodos. Los nodos son sentidos de las palabras o clases abstractas de sentidos, mientras que los enlaces son relaciones semánticas entre los sentidos. Una de las relaciones prototípicas es la que se da entre clase y subclase.

WordNet (Fellbaum, 1998) es una base de datos léxico-conceptual del inglés estructurada en forma de red semántica y construida manualmente. Es el lexicon relacional más completo y extenso existente. La unidad básica en que se estructura es el synset (conjunto de sinónimos), representando un concepto. Los synsets se relacionan entre sí mediante relaciones semánticas básicas, como son la hiponímia/ hiperonímia y la meronímia/holonimia. La versión de WordNet l.5 (WN) contiene 126000 entradas, repartidas entre sinónimos 70%, adjetivos l5% y verbos 10% y la última versión 2.0 tiene un inventario de 144309 palabras.

WN 2.0 ha supuesto una serie de cambios respecto a las variantes anteriores, destinados a aumentar la conectividad en tres direcciones: morfología, desambiguación de glosas y agrupaciones temáticas. El uso de WN en varias tareas o aplicaciones a puesto un grado de polisemia excesivo es decir una granularidad demasiado fina (Palmer, 2000).

Esta herramienta se ha convertido en una referencia para la investigación computacional siendo la más usada en los últimos años, debido fundamentalmente a su cobertura y a su potencial carácter multilingüe a través del proyecto EuroWordNet.

Los lexicones generativos han empezado a ser explotados por la WSD. El uso de los corpus como fuente de información está relacionado con la evolución de la investigación empírica en lingüística. Se puede hablar del análisis manual de textos ya a finales del siglo XIX, pero en lingüística se empiezan a usar a mediados del siglo XX. Los corpus se tratan como fuentes de ejemplos y facilitan el desarrollo de modelos numéricos del lenguaje. El vínculo estrecho con los modelos empíricos explica su período de declive alrededor de los años sesenta, resucitando en los ochenta debido a la aparición de corpus grandes y accesibles a la computadora.

Los corpus son colecciones de textos, las más usadas en el área del PLN, accesibles por computadora, para servir una determinada función y según determinados criterios de acuerdo con un objetivo. Debido a que ofrecen conjuntos amplios de ejemplos para un determinado hecho lingüístico, los corpus permiten el desarrollo de modelos numéricos del lenguaje y en consecuencia el uso de métodos empíricos.

En el área de la WSD, los corpus constituyen una fuente de información en la línea de investigación llamada precisamente WSD basada en corpus. En este caso la desambiguación se realiza mediante un algoritmo que no usa información explícita de una fuente léxica, sino que adquiere conocimientos sobre los sentidos de las palabras a partir de un corpus. La fase de aprendizaje de los algoritmos se puede desarrollar sobre corpus anotados (aprendizaje supervisado) o no anotados (aprendizaje no supervisado).

2.1.1. Corpus no etiquetados

En este caso las unidades léxicas (palabras) no tienen asociadas información lingüística de ningún tipo. Aparecen en los años setenta, y entre los más importantes se encuentran el corpus de *Brown* con un millón de palabras, el *Tresor de Lengue Francase*, y el LOB (Lancaster-Oslo-Bergen).

A partir de los corpus no etiquetados a nivel de sentidos, se pueden inducir automáticamente agrupaciones (clusters) semánticas, como etiquetados de sentidos efectivos (Schustze, 1992). Estas agrupaciones se pueden alinear con inventarios tradicionales de sentidos de los diccionarios. Sin embargo, pueden igualmente funcionar sin esto, sobre todo si se usa para aplicaciones secundarias, como recuperación de información para la cual es más importante la partición de sentidos que la elección de la etiqueta.

2.1.2. Corpus etiquetados

Se incluyen tipos dependiendo de la metodología de anotación, el inventario de sentidos y las correspondientes etiquetas usadas para los sentidos. Un primer tipo son las etiquetados a mano y otra los etiquetados automáticamente.

Entre los corpus etiquetados manualmente más utilizados se hallan SemCor, *line*, *interest* y DSO. La fiabilidad del etiquetado manual, como fuente de referencia en la anotación con sentidos es relativa. Por ejemplo, los corpus DSO y SemCor tienen en común un fragmento del corpus anotado de *Wall Street Journal*, y el etiquetado sobre este fragmento con sentidos de WN, coincide solo en el 57% (Stevenson, 2003).

2.2. Tipos de conocimiento útiles para la WSD

El proceso de la WSD necesita información sobre las palabras ambiguas. Se suelen usar dos principales tipos de conocimiento; sobre el significado sobre los usos de las palabras. Clasificándose en dos categorías:

- Conocimiento independiente del contexto
- Conocimiento dependiente del contexto

Aguirre y Martínez (200l) identifican una serie de clases de información útiles para la WSD:

Categoría sintáctica (*Part of Speech*, PoS), que se usa para organizar los sentidos de las palabras. La categoría preestablecida delimita dentro del conjunto de sentidos asociados a un lemma considerado, el subconjunto de sentidos que corresponden al lema con la categoría especifica. Por ejemplo, en WN *handle* tiene 5 sentidos como verbo (*manejar*) y solo 1 como sustantivo (*asa*).

Morfología, especialmente la relación entre las palabras derivadas y sus raíces. En el proceso de derivación morfológica, algunos sentidos no se trasmiten. Por ejemplo, en WN *agreement* (acuerdo) tiene 6 sentidos y su raíz *agree* (acordar) tiene 7 sentidos.

Colocaciones; suelen tener una base estadística: parque natural, zona verde, etc... Por ejemplo, el sustantivo partido tiene 9 sentidos pero un solo sentido en partido de futbol.

Asociaciones semánticas de las palabras; que se subclasifican de la siguiente manera:

Organización taxonómica; relación básica hipo/hiperonímia. Por ejemplo, silla es-un mueble.

De dominio; se hace referencia a áreas concretas de conocimiento, ejemplo, en el dominio de los deportes *racket* (raqueta) o como *tenis racket* (raqueta de tenis).

Frecuencia de los sentidos. Sobre los cuatro sentidos de *people* (gente), el sentido general corresponde a 90% de las ocurrencias de SemCor.

2.2.1. Información contextual

El contexto es el principal medio para la identificación del sentido de una palabra polisémica, toda labor de WSD se funda en el contexto de la palabra ambigua (*target word*) con el fin de proveer información para su desambiguación (Ide y Veronis, 1998).

Contexto local, consiste en una ventana de unidades léxicas en un texto alrededor de la ocurrencia de la palabra ambigua; varía desde algunas palabras hasta toda la oración. El contexto local ofrece información variada como distancia.

El concepto de distancia está relacionado con el número de palabras (n) que se incluyen en el contexto. Se han dado respuestas muy variadas sobre el valor óptimo de n, es decir, la distancia. Yarowsky (1993) examina varias ventanas y pares de palabras de conjuntos (en –1 y –2 –1 y +1 +1 y +2 respecto a la palabra ambigua). Su conclusión es que el valor óptimo n para la dimensión de la ventana varía con el tipo de ambigüedad. La local de 3 a 4 mientras la relativa de 20 a 50 palabras. En cambio Leacock (1998) propone una ventana local con 3 palabras de clase abierta (sustantivos, verbos, adjetivos y adverbios) en ambas direcciones, derecha e izquierda.

2.3. Clasificación de los métodos de WSD

La polisemia significa que la mayoría de las palabras tienen muchos posibles significados. Un programa computacional no tiene las bases de conocimiento para saber cual es el apropiado, aun cuando esto sea obvio para un humano.

La dificultad de la tarea de WSD explica el desarrollo de una gran variedad de métodos para su solución. Igual que otras tares del PNL, la WSD aplica modelos y técnicas procedentes de otros campos de investigación, principalmente de la Estadística y de la Inteligencia Artificial.

Las propuestas para la solución de esta tarea se clasifican de acuerdo a su principal fuente de conocimiento usado en la elección de sentidos.

Métodos que recurren primero en los diccionarios, tesauros y bases del conocimiento léxico, son conocidos como basados en diccionarios o basados en conocimiento.

Métodos que utilizan información externa y trabajan directamente con textos no anotados se conocen como métodos no supervisados.

Finalmente los métodos supervisados que hacen el uso de textos anotados para su entrenamiento. Aunque usualmente demuestran mejores resultados, requieren de la labor humana para el entrenamiento del clasificador; lo que se considera poco deseable. Ambos tipos métodos adoptan sus nombres de le terminología de ML

2.4. Métodos basados en diccionarios

Estos métodos dependen principalmente de diccionarios o textos sin anotar, sus principales recursos son:

- Diccionarios accesibles por computadora (MRD)
- Corpus sin marcar

Teniendo como objetivo a desambiguar todas las palabras de clase abierta. Donde los diccionarios proveen por cada palabra información como: lista de significados, definiciones y ejemplos típicos. Un ejemplo de este conocimiento es WN.

2.4.1. Método de Lesk

Identifica los sentidos de una palabra en el contexto usando el solapamiento de las definiciones (Lesk, 1986).

Algoritmo

1. Regresar de un MRD todos los sentidos de las palabras a desambiguar.

2. Determinar el solapamiento de definiciones para todas las posibles combinaciones de sentidos.

3. Escoger el sentido con el mayor solapamiento

Pero cuando el número de palabras es mayor. Por ejemplo, 9 palabras la combinación de sentidos es de 43,929,600.

2.4.2. Metodo de Cowie y Guthrie,

Solucionando el problema combinatorio del metodo de Lesk por (Cowie y Guthrie, 1992). Utilizando recocido simulado (*Simulated annealing*), que simula en enfria-

miento de un cristal. Definiendo la función E = combinación de sentidos en un texto dado, y encontrar la combinación de sentidos con el solapamiento mayor.

Algoritmo

1. Comenzar con E = el sentido más frecuente por cada palabra

2. En cada iteración, reemplazar el sentido de una palabra escogida de manera aleatoria en el conjunto con un sentido diferente y medir E

3. Detener las iteraciones cuando no hay cambios en la configuración de sentidos

Otra solución al problema combinatorio del algoritmo de Lesk (Kilgariff y Rosenweig, 2000) es una versión simplificada en la cual se identifica el sentido correcto de una palabra a la vez, al contrario del algoritmo original, reduciendo significativamente el espacio de búsqueda.

2.4.3. Selectional preferences.

Otro modo de restringir los posibles significados de las palabras en un contexto dado es capturando información a cerca de las posibles relaciones entre clases semánticas.

Ejemplo.

Wash a dish	*Cook a dish*
Lavar-objeto	cocinar-objeto

Resolviéndose este problema, mediante el conteo de frecuencias, medidas de la teoría de la información y relaciones entre clases.

2.4.4. Similitud semántica.

Las palabras en un discurso deben de estar relacionadas en significados, para la coherencia del discurso (Haliday y Hassan, 1976). Usando esta propiedad la WSD identifica significados para las palabras que comparten un significado común.

La similitud se determina entre dos pares de conceptos o entre la palabra y su contexto, dependiendo de métricas de similitud en redes semánticas (Mihalcea *et al.,* 1989) para realizarlo.

Estas métricas como entrada reciben 2 conceptos (mismos PoS) y como salida generan una medida de similitud, ejemplos de estas métricas son la de (Leacock y Chodorow, 1998) y (Resnik, 1995).

2.5. Métodos basados en heurísticas

2.5.1. Sentido más frecuente

La heurística del sentido más frecuente**,** identifica el sentido más usado y el uso de ese significado.

Ejemplo, **planta** es más usada como flora que como fábrica.

Existen dos propuestas para estos métodos:

- Encontrar el sentido más frecuente en un corpus anotado.
- Encontrar el sentido más frecuente usando una distribución de similitud (MacCarthy *et al*., 2004).

2.5.2. Un sentido por discurso

Otra heurística es la de **un sentido por discurso**, donde las palabras tienden a preservar su significado a través de todas sus ocurrencias en un discurso dado (Gale, Church y Yarowsky, 1992), lo que significa que si la palabra *planta* ocurre 10 veces en un discurso todas las instancias de *planta* tienen el mismo sentido.

2.5.3. Un sentido por colocación

Todas las palabras tienden a preservar el sentido cuando son usadas en la misma colocación (Yarowsky, 1993), siendo este método fuerte en palabras adyacentes y débil conforme la distancia entre las palabras aumenta. Ejemplo, planta industrial

2.6. Métodos supervisados

Colectan un conjunto de ejemplos que ilustran las posibles clasificaciones de un evento. Identificando patrones en los ejemplos y asociándolos a una clase en particular, para después generalizar estos patrones en reglas y finalmente aplican estas reglas a clasificar un nuevo evento.

Definiéndose esta tarea como, un método que induce una clasificación con texto anotados manualmente con sentidos utilizando técnicas de ML. Donde sus recursos son:

- Texto anotado con sentidos
- Diccionarios (como inventario de sentidos)
- Análisis sintáctico (PoS tagger, parser)

Teniendo como objetivo desambiguar una palabra por contexto, reduciendo la WSD a un problema de clasificación donde la palabra ambigua es asignada al más apropiado sentido dado un conjunto de posibilidades basadas en el contexto en el cual ocurre.

Una vez preparados los datos, cualquier algoritmo supervisado puede ser utilizado. Muchos de estos algoritmos que han sido utilizados con buenos resultados son:

- Support Vector Machines
- Nearest Neighbor Classifier
- Decision Trees
- Naïve Bayesian Classifier
- Redes Neuronales

2.6.1. Naïve Bayesian Classifier

Por ejemplo, el Naïve Bayesian Classifier, que es bien conocido entre la comunidad de ML que este algoritmo tiene buen desempeño a través de un rango amplio de tareas (Domingos y Panzzani, 1997), donde la WSD no es la excepción. Usa una independencia condicional a través de los rasgos, dados los sentidos de una palabra. La forma del modelo es asumida, pero los parámetros son estimados de las instancias de entrenamiento.

Cuando es aplicado a la WSD, los rasgos son regularmente una "bolsa de palabras" que proviene de los datos de entrenamiento. Usualmente se utilizan miles de rasgos binarios para indicar si una palabra esta presente en el contexto de la palabra ambigua o no.

Basando sus decisiones en la inferencia bayesiana, dando unos rasgos observables, ¿Cuál es el sentido más parecido?, estimando la probabilidad de que los rasgos observables de un sentido dado y estimando la probabilidad condicional del sentido.

En estudios comparativos el Naïve Bayesian Classifier tuvo un mejor desempaño que otro métodos (Leacock et. al, 1993), compara el método con una red neuronal y (Pedersen, 1998) lo compara con un árbol de decisión (*Decision Trees*).

2.7. Métodos no supervisados

Los métodos no supervisados identifican patrones en una gran muestra de datos, sin el beneficio de anotaciones manuales.

Estos patrones son usados para dividir los datos en clusters, donde cada miembro del cluster tiene más en común que con otros miembros de otros clusters.

Dando un resultado interesante cuando las etiquetas de sentidos son removidas de los datos supervisados, uno puede encontrar diferencias en las mis más clases del aprendizaje supervisado.

Los métodos supervisados identifican características o rasgos y los no supervisados encuentran similitud entre contextos.

Definiéndose la tarea, en métodos que agrupan palabras basadas en la similitud del contexto. Con hipótesis como (Miller y Charles, 1991); palabras con significados similares tienden a ocurrir en contexto similares. Y su recurso principal es el uso de corpus muy grandes (Ej. Internet).

Teniendo como ámbito el desambiguar una palabra por contexto. Reduciéndose el problema a la discriminación de sentidos, encontrando palabras ambiguas que ocurren en contextos similares y ponerlos en un cluster.

2.7.1. Método de Yarowsky

Por ejemplo, las palabras no solo tienden a ocurrir en colocaciones que identifican su sentido, tienden a ocurrir en múltiples colocaciones. Esto produce un método para un etiquetador de sentidos (Yarowsky, 1995). Si uno comienza con un pequeño conjunto semilla de ejemplos representativos de los sentidos de una palabra uno puede incrementalmente aumentar la semilla con ejemplos adicionales de cada sentido, usando la combinación de tendencias un sentido por discurso y un sentido por colocación.

2.8. Discusión y crítica de los métodos existentes

Los métodos supervisados hacen el uso de textos anotados para su entrenamiento. Aunque usualmente demuestran mejores resultados, requieren de la labor humana para el entrenamiento del clasificador; lo que se considera poco deseable.

Los métodos no supervisados tienen el potencial de superar el cuello de botella en la adquisición del conocimiento que se da en los métodos supervisados por lo escaso y costoso de los textos anotados.

Por su parte los métodos basados en conocimiento dependen del inventario de sentidos y en el caso del algoritmo de Lesk se crea un problema combinatorio cuando el número de palabras aumenta.

2.9. WSD software

Algunos ejemplos de sistemas para la WSD, tanto métodos supervisados como no supervisados así como herramientas para el ML y sus direcciones en Internet.

Duluth Senseval-2 systems. Basado en árboles de decisión, participante en Senseval-2 y 3; www.d.umn.edu/~tpederse/senseval2.html.

SyntaLex. Versión mejorada del Duluth Senseval-2 con el uso de rasgos sintácticos, participó en Senseval-3; www.d.umn.edu/~tpederse/syntalex.html.

WSDShell. Shell para correr experimentos en Weka (ML); www.d.umn.edu/~tpederse/wsdshell.html.

SenseTools. Para la fácil implementación de WSD, usado por los tres sistemas anteriores. Transforma datos del tipo Senseval-formatted en archivos para Weka; www.d.umn.edu/~tpederse/sensetools.html.

SenseRelate::TargetWord. Identifica los sentidos de una palabra basado en la similitud semántica con sus vecinos; search.cpan.org/dist/WordNet.

SenseRelate-TargetWord Usa WordNet::Similarity –mide la similitud en WordNet; search.cpan.org/dist/WordNet-Similarity.

SenseLearner Método mínimamete supervisado. http://lit.csci.unt.edu/~senselearner

InfoMap Representa los significados de las palabras en un vector (no supervisado). http://infomap-PLN.sourceforge.net

SenseClusters Encuentra clusters de palabras que ocurren en contextos similares (no supervisado). http://senseclusters.sourceforge.net

2.10. Evaluacion

La calidad de los resultados es fundamental para la WSD como para toda tarea del PNL, con la cual la evaluación de los sistemas de desambiguación es un problema central en el área. En el caso específico de la WSD, la comparación de los resultados obtenidos en diversas evaluaciones se ha visto dificultada por las muchas variables involucradas en el proceso: medias de evaluación, material de entrenamiento, granularidad de sentidos, etc.

Stevenson y Wilks (2001) consideran que la evaluación en el campo de la WSD es distinta de otras tares del PLN, en diferentes aspectos: La cobertura de los sistemas (un conjunto de palabras vs. Vocabulario no restringido), la variedad y la multiplicidad topológica de la información explotada por cada tarea de desambiguación.

La calidad de un sistema de WSD se suele cuantificar mediante dos medias de precisión, la precisión absoluta y la precisión relativa, provenientes del área de IR (Van Rijsbergen, 1971, Salton y MacGill, 1983, Stevenson, 2003).

La precisión absoluta (*recall***, R**) es la métrica básica para decidir la calidad de la tarea de WSD porque muestrea el numero de casos correctamente desambiguados sobre todos los casos de prueba.

La precisión relativa (*precision***, P**) se define como el porcentaje de respuestas correctas sobre todas los casos tratados por el sistema de WSD. Esta medida favorece a los sistemas que obtienen una alta fiabilidad en la asignación de sentidos.

Las métricas son realmente informativas si se interpretan conjuntamente. Un sistema puede tener una precisión relativa muy buena, pero una precisión absoluta muy baja, lo que significa que ha identificado la respuesta correctamente para muy pocos caos, pero con alta fiabilidad. En general hay un equilibrio entre las dos medidas y su combinación se puede expresar mediante la métrica F *(F-measure*) según la formula:

$$F_{\alpha} = (1+\alpha)\frac{\text{precisión} \times \text{recall}}{\alpha \times \text{precisión} + \text{recall}}$$

Donde es variable. La importancia relativa de las dos medidas P y R en la métrica F se puede cambiar mediante la variación de , usualmente se prestablece a 1 (Stevenson, 2003).

Ejemplo:

- Palabras desambiguadas correctamente **50**
- Conjunto de prueba 100 palabras **Recall** = 50 / 100 = **0.50**
- Sistema intenta 75 palabras **Precision** = 50 / 75 = **0.66**

2.11. Senseval

En 1997, se sentaron las bases de una competición libre y voluntaria, denominada Senseval (SENSe EVALuation, http://www.senseval.org), con el propósito de explorar los aspectos científicos y técnicos de la WSD y así poder establecer unas bases objetivas para la evaluación de estos sistemas.

El primer certamen Senseval tuvo lugar en 1998 y las lenguas participantes fueron el inglés, el francés y el italiano. La metodología desarrollada permitía evaluar los sistemas de desambiguación exclusivamente en función de una muestra léxica, es decir, había que determinar automáticamente el sentido de una única palabra en un contexto determinado. Con Senseval -2, las lenguas participantes se incrementaron hasta un total de 12 (inglés, francés, italiano, español, vasco, danés, sueco, holandés, estonio, checo, chino y japonés) al igual que también se incrementaron el tipo de tareas posibles en las que podían concursar los distintos participantes. En este sentido, se diseñaron tres tipos de tareas que básicamente se diferenciaban entre sí por el número de palabras que los programas tenían que desambiguar. Con la finalidad de entender mejor la metodología que se sigue en Senseval, se presenta a continuación de manera más detallada en qué consiste cada una de estas tareas.

2.11.1. Tipos de tareas

En Senseval-2 se diseñaron tres tipos de tareas para la evaluación de los sistemas de desambiguación: tarea basada en una muestra léxica (*lexical sample task*), tarea léxica completa (*all-words task*) y tarea de traducción (*translation task*).

La tarea basada en la muestra léxica sólo se fija en una única palabra por frase, mientras que en la tarea léxica completa, los sistemas tienen que desambiguar semánticamente todas las palabras, exceptuando las funcionales (conjunciones, preposiciones, artículos, etc.), que aparecen en las frases de los corpus seleccionados. La tarea de traducción, en la que sólo ha participado la lengua japonesa, es de hecho un subtipo de muestra léxica porque sólo hay que desambiguar una única palabra; la diferencia es que el sentido de la palabra se define de acuerdo a su traducción.

En definitiva, en la tarea léxica completa se evalúa la capacidad del sistema automático de desambiguar cada una de las palabras (u ocurrencias) que aparecen en el corpus que se proporciona. En las tareas basadas en una muestra léxica se evalúa a los sistemas en función de la desambiguación de las diferentes ocurrencias de una muestra de palabras previamente seleccionadas en el contexto de una frase.

En la Tabla 1 se muestra de manera esquemática las tareas en las que han participado las distintas lenguas y el número de equipos participantes (entre paréntesis):

Tabla 1. Tipos de tareas realizadas en Senseval

Tareas	
Tarea basada en una muestra léxica	Español (5), japonés (8), Vasco (2), coreano (2), Inglés (15), sueco (5), Italiano (2)
Tarea de traducción	Japonés (8)
Tarea léxica completa	Checo (1), estonio (2), Holandés (1), inglés (12)

Para poder llevar a cabo dichas tareas es necesario disponer de dos tipos básicos de datos: un diccionario (o léxico) y un corpus etiquetado manualmente que servirá como *gold standard*, y a partir del cual se creará el corpus de entrenamiento y el corpus de evaluación. Todos los sistemas que concursan tienen que analizar semánticamente el mismo corpus y, en consecuencia, es necesario que cada lengua participante disponga de este corpus y de un diccionario en el que se incluyan los distintos significados de las palabras que se deben desambiguar.

2.11.2. Desempeño

El desempeño alcanzado por los sistemas del estado del arte (*state-of-the-art*) de WSD, alcanzan una exactitud del 95%, usando poco conocimiento de entrada: por ejemplo (Yarosky 1995) uso un método semi-supervisado para evaluar 12 palabras (96.5%), y (Stevenson y Wilks 2001) usaron *part-of-speech* (y otras fuentes de conocimiento) en todas las palabras de un texto usando un diccionario (94.7%).

En los sistemas de WSD la exactitud en la polisemia general ha sido más difícil de alcanzar, pero se ha mejorado conforme avanza el tiempo. En 1997, Senseval-1 (Kilgariff y Palmer) se encontró una exactitud del 77% en la tarea (*English lexical sample task)*, justamente debajo del 80% del desempeño humano de los anotadores *(Inter-tagger agreement*); auque se ha estimado que el desempeño humano es del 95%.

En 2001 los resultados en el Senseval-2 parecieron menores, pero con una tarea más difícil a desarrollar, el uso de los sentidos de grano fino de WordNet. El mejor sistema alcanzó un 64% (anotadores 86%). Mostrando que los métodos supervisados tuvieron el mejor desempeño, contrastando con los no supervisados que obtuvieron un 40%. Donde el sistema base (*baseline*) del sentido más frecuente fue de 48% y la elección de sentidos de manera aleatoria de 16%.

Para el 2004, los mejores sistemas del Senseval-3 alcanzaron un desempeño, para métodos supervisados del 71.8% con un 67% de los anotadores (no expertos) y para los no supervisados un 66% de exactitud.

Y en el 2007, SemEval, la evaluación de la tarea se realizó aplicándola a la IR, alcanzando el mejor sistema supervisado 75.39% siendo este un sistema basado en el conocimiento.

3. Marco teórico

La tarea de medir la cohesión entre palabras proviene de la recuperación de información. La bien conocida medida estadística para el cálculo en los números de ocurrencias de palabras y sus combinaciones es la información mutua (MI:*Mutual Information*) apropiada para evaluaciones en corpus.

La Web es inmensa, gratis y disponible con un solo clic. Esta contiene cientos de billones de palabras en textos y pueden ser usadas para todo tipo de investigación.

Una manera simple de su utilidad es la corrección ortográfica. Por ejemplo, "hacer" o "haser" Google da 45,200,000 para el primero y 337,000 para el segundo (después sugiriendo lo que se trato de decir). Respondiendo a la pregunta de cual palabra está correctamente escrita.

El rápido desarrollo de la Web, invita a revisar que métodos se encuentran disponibles para los textos. En los buscadores de Internet, los datos estadísticos se encuentran medidas en función del número de páginas relevantes donde se encuentran las ocurrencias de la palabra.

Cada texto es una secuencia de formas de palabras (*word forms*). Por ejemplo, cadenas de letras de un delimitador al siguiente. Las formas de las palabras con significados comunes son asociadas en lexemas. Dividiéndose a los lexemas en 3 categorías:

- *Content word*: sustantivos; adjetivos; adverbios; verbos exceptuando los auxiliares.
- *Functional words*: preposiciones; conjunciones y verbos auxiliares.
- *Stop words*: pronombres; nombres propios exceptuando los lugares geográficos o económicos bien conocidos reflejados en diccionarios y enciclopedias; así como otras partes del habla (PoS).

Considerando las ocurrencias y coocurrencias de combinaciones de palabras en un texto a una limitada distancia entre ellas. Finalmente su coocurrencia puede considerarse estable, si su frecuencia relativa (probabilidad empírica) $N(P_1,P_2)/S$ de la ocurrencia de P_1 y P_2 es mayor que el producto de sus frecuencias relativas $N(P_1)/S$ y $N(P_2)/S$ de los componentes tomados a parte (S es el tamaño del corpus).

Usando logaritmos, se tiene el criterio de cohesión entre palabras conocido como información mutua.

La MI tiene una importante característica de escalabilidad: si los tamaños de todos los bloques de construcción S, $N(P_1)$, $N(P_2)$ y $N(P_1,P_2)$ son multiplicados por el mismo factor positivo, MI conserva su valor.

Otro criterio diferente del de MI pero que incluye los mismos bloques de construcción es el de *scalable Pearson Correlation Coefficient* o el de *non-scalable Association factor*.

Cualquier buscador de Internet automáticamente regresa estadísticas acerca de la palabra consultada o una combinación de palabras medidas en función de páginas relevantes, y ninguna información de las ocurrencias de palabras o co-ocurrencias esta disponible. Se puede reconceptualizar MI usando todos las N() como numero de paginas relevantes y S como el total de paginas que maneja el buscados (Bolshakov y Bolshakova, 1999).

Ahora N()/S no son probabilidades de eventos relevantes: las palabras que ocurren en la misma pagina son indistinguibles en las estadísticas, siendo contadas sólo una vez, mientras que la misma página es contada repetidamente por cada palabra incluida. Este criterio de cohesión de las palabras he llamado *Stable Connection Index*.

Utilizando una modificación de la frecuencia relativa en los métodos propuestos para medir la cohesión entre las coocurencias de palabras que un buscador Web (utilizado como corpus) nos puede regresar dada una consulta.

4. Métodos propuestos

Como Internet contiene la mayor colección de textos almacenados de manera electrónica, se utilizo a la Web como corpus para la WSD. Proponiéndose para ello 2 algoritmos par al solución de esta tarea. Utilizando como inventario de sentidos WordNet 2.1.

4.1. Algoritmo 1

Para la mejor explicación de este algoritmo, se proveen los pasos del mismo así como un ejemplo de su ejecución.

Entrada: Palabra ambigua y contexto no etiquetado a nivel de sentidos.

Salida: Palabra etiquetada a nivel de sentidos.

Procedimiento*:*

1. Seleccionar un conjunto de sustantivos alrededor del sustantivo a desambiguar w usando una ventana de tamaño 3 (este conjunto será llamado C).

2. Por cada sentido w_k de w, por cada sinónimo S_{ik} de w_k, calcular $f(C,S_{ik})$ y $f(S_{ik})$.

3. Asignar a cada sentido w_k un peso dependiendo de la función $F(w_k,C)$ que combina los resultados obtenidos en el paso anterior.

4. Seleccionar el sentido que tenga el peso mayor.

Donde $f(C,S_{ik})$ es la frecuencia de ocurrencias del sinónimo en el contexto, y $f(S_{ik})$ es la frecuencia del sinónimo, tomados de la Web

$$F(w_k,C)= f(C,Si_k)/ f(S_{ik}). \quad (1)$$

Donde (1) es la sumatoria de probabilidades.

Ejemplo

El tipo de consulta utilizado es el de “palabra + palabra + … + palabra”, donde un buscador Web revisará si esas palabras se encuentran en el mismo documento, no necesariamente que se encuentren juntas en el texto, y regresará la coocurrencia de ellas, el número de documentos donde se encuentran. Este ejemplo fue tomado al azar del corpus de Semcor.

Entrada:

palabra= investigation

contexto= Friday investigation Atlanta primary_election evidence

Paso 1

Seleccionar un conjunto de sustantivos alrededor del sustantivo a desambiguar w usando una ventana de tamaño 3.
C={Friday, investigation, Atlanta, primary_election, evidence}

Paso 2

Por cada sentido w_k de w, por cada sinónimo s_{ik} de w_k, calcular $f(C,Si_k)$ y $f(S_{ik})$.

Sentido 1 w_1 = investigation

Sinónimo 1 s_1= probe

Frecuencia = 75000000

Contexto = Friday probe Atlanta primary_election evidence

Frecuencia=155000

Sinónimo 2 s_2= investigation

Frecuencia = 157000000

contexto = Friday investigation Atlanta primary_election evidence

Freecuencia= 487000

Sentido 2 w_2= investigation

Sinónimo 1 s_1=investigation

Frecuencia =157000000

contexto =Friday investigation Atlanta primary_election evidence

Frecuencia = 487000

Sinónimo 2 s_2= investigating

Frecuencia = 32600000

contexto = Friday investigating Atlanta primary_election evidence

Frecuencia = 496000

Paso 3

Asignar a cada sentido w_k un peso dependiendo de la función $F(w_k,C)$ que combina los resultados obtenidos en el paso anterior.

$F(w_k,C)= f(C,S_{ik})/ f(S_{ik})$.

$F(w_1,C)= 0.005$

$F(w_2,C)= 0.01$

Paso 4

Seleccionar el sentido que tenga el peso mayor.

ELECCION= w_2 Sentido 2

4.2. Algoritmo 2

Este algoritmo es una variante del algoritmo de Lesk simple (Satanjeev y Pedersen, 2002), donde la medida de similitud no es el solapamiento de glosas sino la co-ocurrencia de la glosa con el contexto en la Web. Se eligió Lesk simple por la reducción del problema combinatoria y el uso de las definiciones también reduce el

problema de complejidad del algoritmo anterior, cuando existen muchos sinónimos en un sentido, en vez de preguntar en Internet por cada sinónimo solo se pregunta por la definición de cada sentido.

Entrada: Palabra ambigua y contexto no etiquetado a nivel de sentidos.

Salida: Palabra etiquetada a nivel de sentidos.

Procedimiento:

1. Seleccionar un conjunto de sustantivos alrededor del sustantivo a desambiguar w usando una ventana de tamaño 3 (este conjunto será llamado C).

2. Por cada sentido w_k de w, y la glosa G de w_k, calcular f(C,G) y f(G).

3. Asignar a cada sentido w_k un peso dependiendo de la función $F(w_k,C)$ que combina los resultados obtenidos en el paso anterior.

4. Seleccionar el sentido que tenga el peso mayor

$$F(w_k,C)= f(C,G)/ f(G). \quad (2)$$

5. Análisis diseño e implementación

Se decidió el uso de RUP (*Rational Unified Process*) porque proveerá de una aproximación para la asignación de tareas y responsabilidades junto con una organización para el desarrollo del proyecto.

5.1. Análisis

Siguiendo el desarrollo de software iterativo, el manejo de requerimientos y una arquitectura basada en componentes. El proceso se describiría a lo largo de dos ejes:

- Horizontal, representa el tiempo, en términos de fases, iteraciones y ciclos.
- Vertical, representado por actividades, artefactos y *workflow*

Comenzando con la primera iteración, la fase de **Inicio (***Inception***)**. Donde se delimitan los alcances del proyecto. Identificando las entidades externas, como interactúan con el sistema. Envolviendo la identificación de casos de uso y una pequeña descripción de los más significativos.

Los primeros artefactos a realizar en esta iteración serán el diagrama de bloques del sistema completo (Ej. Tokeniser + Gazzetter + etc.), como del modulo de desambiguación en especifico. Así como el diagrama de casos de uso y un posible modelo del diagrama de clases y secuencias.

Para dar paso a las actividades que consisten en la revisión de los mismos, mejorándolos y recavando la información necesaria para la segunda iteración **(Elaboración)**, donde en la implementación se realizará con un modelo orientado a objetos. En la iteración de (**Construcción**) se probará el sistema en una aplicación del PLN.

El sistema será visto como una caja negra en la cual la entrada es un documento con formato en texto plano y su salida un documento con etiquetado de sentidos. Por ejemplo, Figura 1., listo para ser utilizado por cualquier aplicación del PLN.

Fig. 1. Arquitectura del sistema

Es objetivo de este trabajo terminal es desarrollar un método para WSD no supervisado usando sinónimos y las palabras semánticamente cercanas a la palabra dada.

Específicamente, nos interesa qué aportación a la desambiguación puede dar cierta información específica de la palabra, sin usar otra información que quizá mejoraría el desempeño pero no es de nuestro interés en este trabajo. Con eso, no es la meta el superar a los mejores desambiguadores existentes, sino sí el superar los desambiguadores que sólo usan el mismo tipo de información que consideramos en este trabajo terminal.

5.1.1. Diagrama de bloques

El sistema funciona con una arquitectura en *pipeline*, mostrado en la Figura 2, donde los módulos se llaman uno después de otro para que al final el desambiguador realice el etiquetado con sentidos.

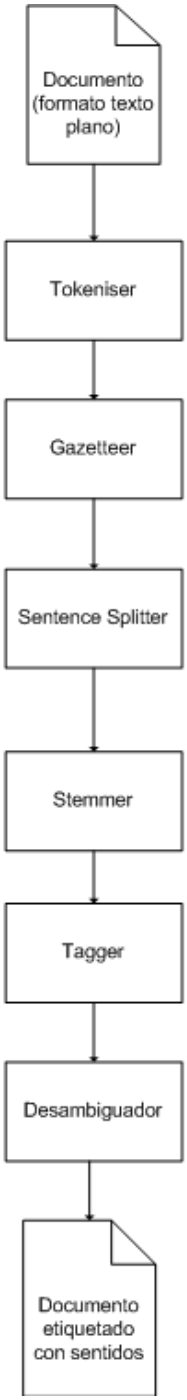

Fig.2. Diagrama de bloques del sistema

5.1.2. Diagramas de caso de uso

La Desambiguación automática del sentido del las palabras ha sido una tarea intermedia (Wilks y Stevensen, 1996) y no es un fin en si misma, pero es necesaria para completar algún nivel del lenguaje. Es obviamente esencial para aplicaciones como:

- Traducción automática (ejemplo: Traducir *.bill*. del ingles al español. ¿Es un pico o una cuenta?),
- Recuperación de información (ejemplo: Encontrar todas las páginas Web acerca de. *cricket*.. ¿El deporte o el insecto?).
- Question Answering (QA) (ejemplo: ¿Qué posición tiene *.George Miller*. acerca del control de armás? ¿El sicoanalista o el congresista?) y casi todas las tareas de procesamiento de texto.

Como se ve en la Figura 3, la interacción del sistema con algunas aplicaciones o programas externos que serán los actores del sistema.

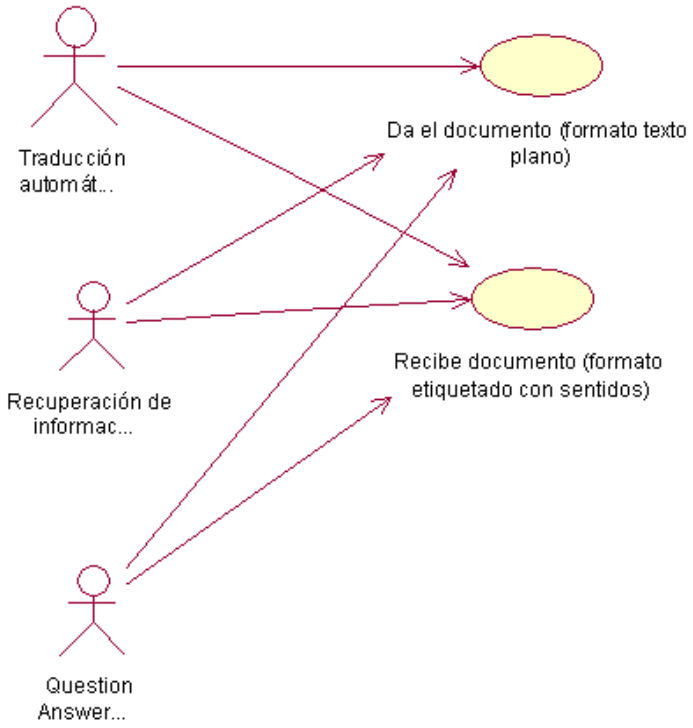

Fig. 3. Caso de uso sistema

En la Figura 4 se detallan los requerimientos del modulo desambiguador. El diagrama de casos de uso nos muestra los sistemas con el que interactúa el modulo. Donde el en actor del diagrama es una aplicación, esta dará un documento que necesite desambiguar y el sistema le regresará el documento etiquetado con sentidos de WordNet 2.1.

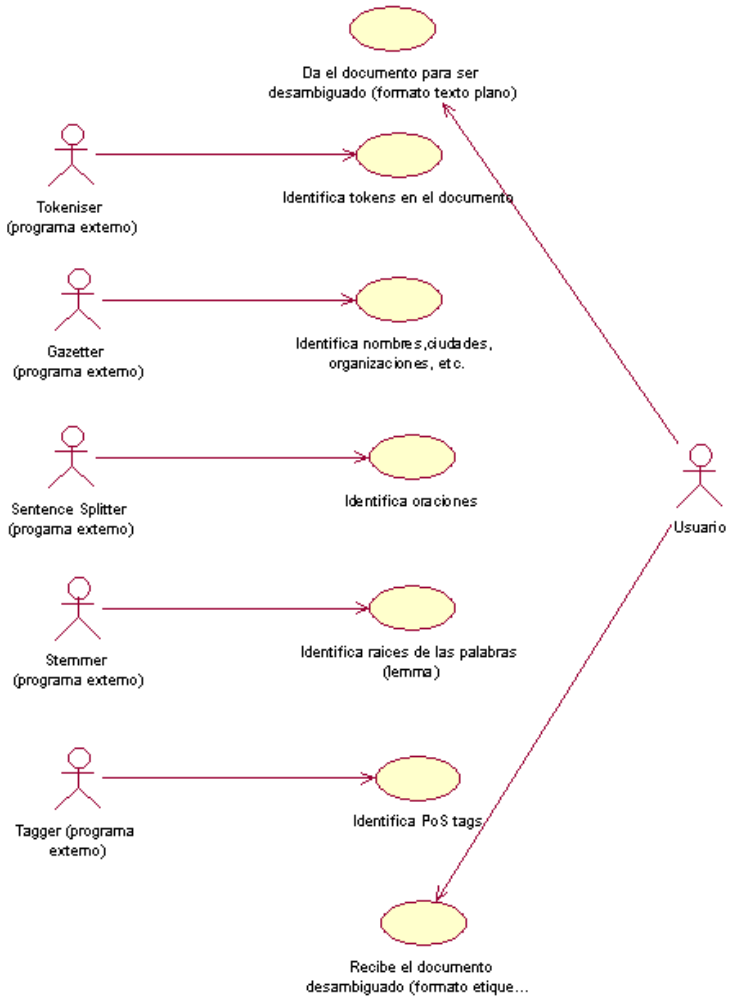

Fig. 4. Caso de uso desambiguador

5.1.3. Diagramas de clases

Para la correcta ejecución e interacción con una aplicación se han identificado 2 tipos de recursos

- **Recursos de procesamiento**: entidades como Tagger, Tokeniser, modulo desambiguador, etc.
- **Recursos visuales**: Componentes para la visualización y edición (GUI) para la interacción con el usuario (alguna aplicación Ej. Traducción automática).

Aquí se muestran los diagramas de clases de los recursos de procesamiento. Los recursos de procesamiento son controlados por un objeto del mismo nombre, que supervisara el orden de ejecución de los módulos, Figura 5, así como la interacción con los recursos visuales, mostrando cada etapa del procesamiento así como el resultado final.

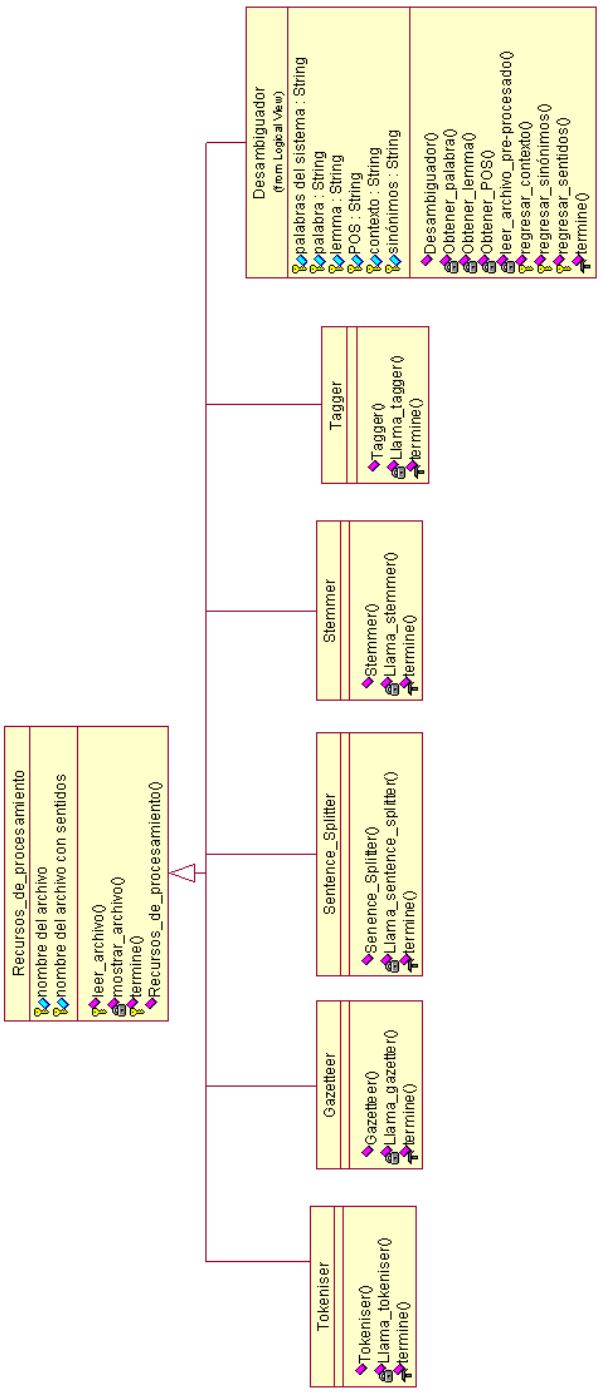

Fig. 5. Recursos de procesamiento

Como se muestra en la Figura 6. El diseño orientado a objetos del modulo desambiguador**.**

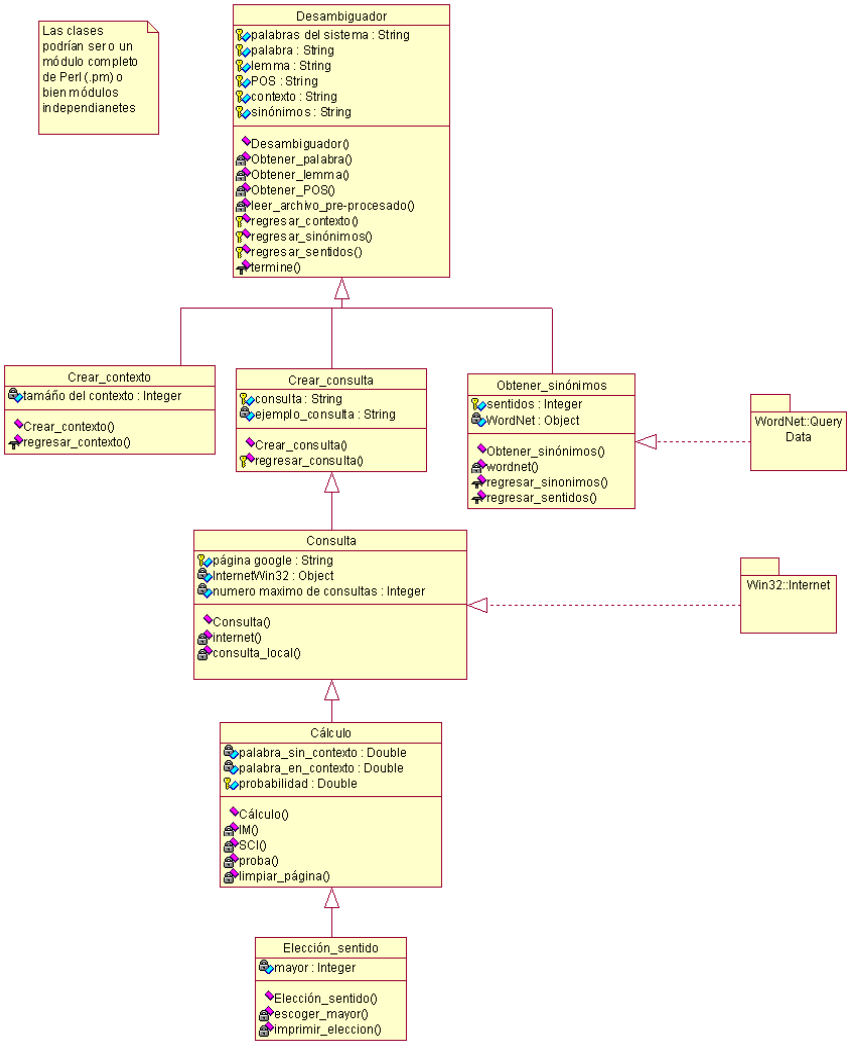

Fig. 6. Clases desambiguador

5.1.4. Diagramas de secuencia

Aquí se puede ver como el objeto recursos de procesamiento llama a cada uno de los módulos y el orden necesario de los mismos, Figura 7.

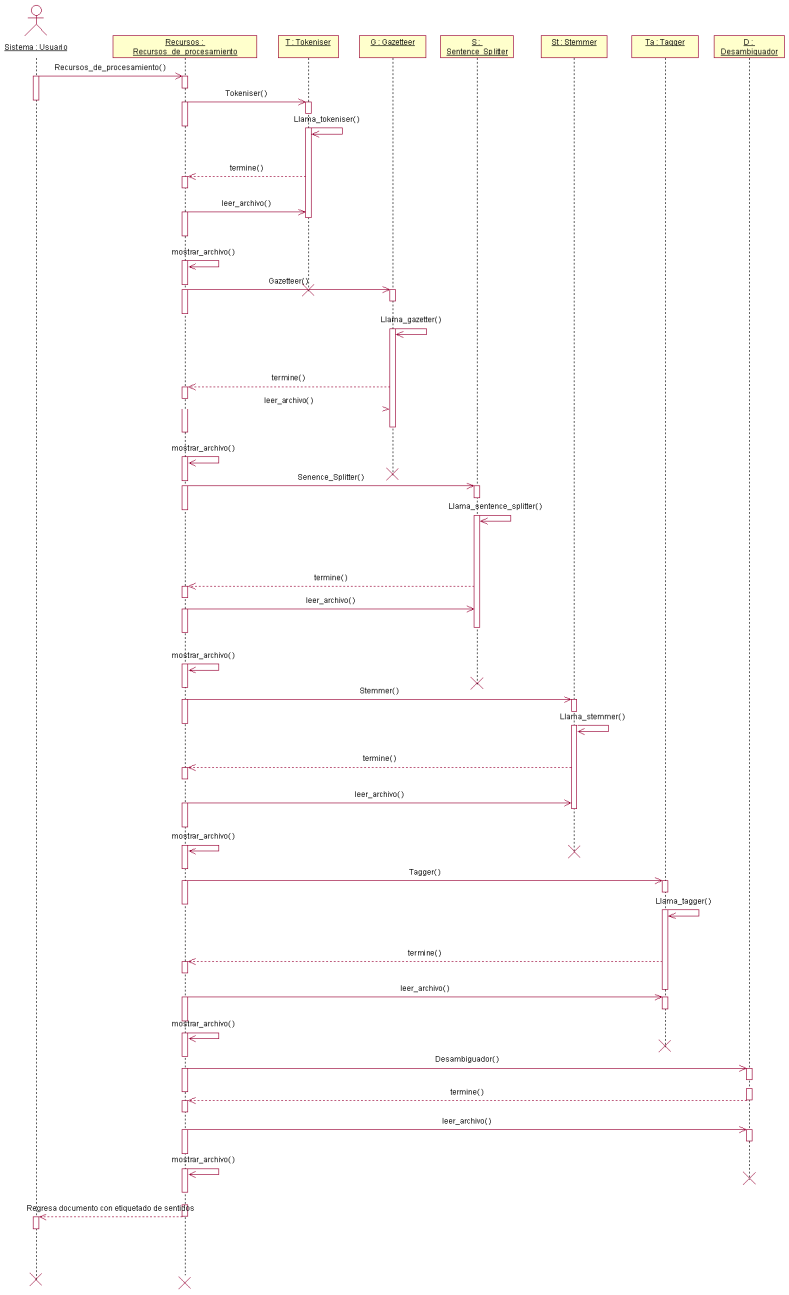

Fig. 7. Diagrama de secuencias del sistema.

El modulo desambiguador también necesita una correcta y ordenada ejecución para llegar al resultado final. Y este control es llevado a cabo por el objeto desambiguador. Figura 8.

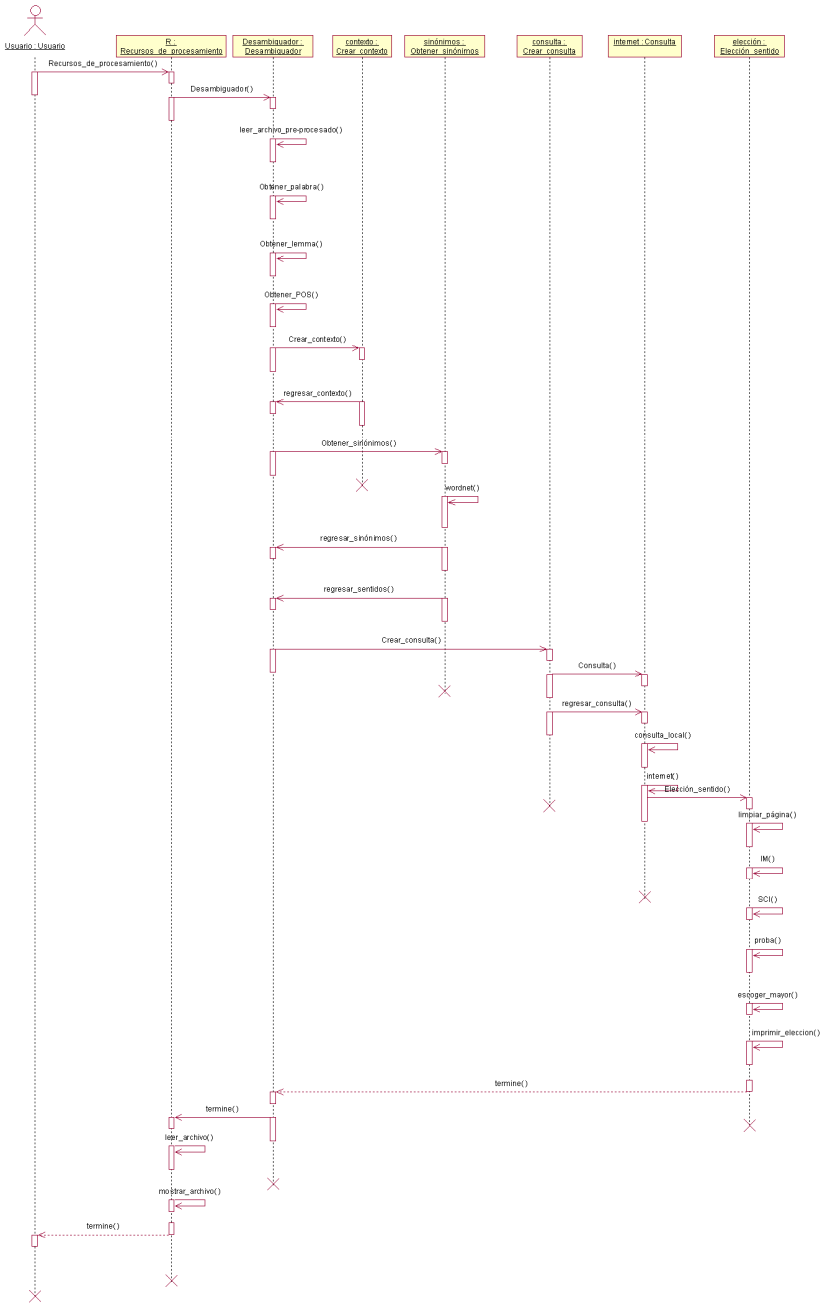

Fig. 8. Diagrama secuencias modulo desambiguador

5.2. Implementación

La traducción automática fue una de las primeras aplicaciones imaginadas para las computadoras. Por primeras vez mostrado por IBM en 1954 con un sistema básico de traducción palabra por palabra.

Siendo la MT comercialmente interesante, por ejemplo, para los Estados Unidos que han investigado la MT por motivos de inteligencia. También es una de las características más usadas de Google y la Unión Europea gasta más de 1000, 000,000 de Euros al año en gastos de traducción. Donde una semi–automatización del proceso podría darles grandes ahorros.

Por el lado académico, la MT requiere de otras tecnologías del PLN. Potenciando la traducción por la WSD, el parseo, *name entity recognition*, etc.

Lo que hace tan difícil a la MT es el orden de las palabras, por ejemplo, en inglés el orden de una oración es sujeto–verbo–objeto y en japonés el orden es sujeto– objeto–verbo. La ambigüedad de las palabras y otros como pronombres e *Idioms.*

Dándose diferentes propuestas para la solución de esta tarea como la traducción palabra por palabra, la transferencia sintáctica, los métodos *interlingua* y la traducción estadística.

Para este trabajo terminal se opto por implementar el algoritmo que utiliza sinónimos, mostrado en el capitulo anterior para potenciar la MT en un esquema de traducción palabra por palabra. Usando un MDR bilingüe, en este caso EuroWordNet para traducir cada palabra (encontrada en el diccionario) en el texto.

Como ventajas se encuentra la facilidad de implementación con resultados que dan una idea general de lo que trata el texto y desventajas como el orden de las palabras que reducirá la calidad de traducción.

5.2.1. Diseño general del sistema

La implementación del algoritmo se realizo principalmente en dos fases: la primera donde se eligió que programas relazarían el preprocesameinto y la segunda donde el módulo desambiguador fue acoplado al pipeline para aceptar la entrada del preprosesamiento. Para finalizar con la aplicación final de MT, de español a inglés y de inglés a español.

La fase de preporcesamineto es realizada por el conjunto de herramientas proporcionadas por FreeLing (Atserias *et al*. ,2006). Que es una librería de código abierto que provee servicios básicos de PLN como (Tokenizer, análisis morfológico, PoS

tagging, etc). Donde cada programa será llamado desde los recursos de procesamiento descritos en la sección anterior.

La elección de esta librería radica en la cobertura de la misma de varios idiomas como español, inglés, italiano. Otra ventaja de esta librería es el uso de una única función *analyzer* que permite ver todo como una solo aplicación y no como software separado, así como la codificación en C++ que le da ventaja sobre otras arquitecturas, en términos de velocidad de ejecución y facilidad de uso, por ejemplo, Gate (Bontcheva *et al*., 2004) plataforma desarrollada en Java la cual tiene la desventaja de solo ser compatible con otros módulos escritos en Java, teniendo también por objetivo al dividir el sistema en módulos, buscar independencia de los mismos en función de los lenguajes de programación.

Un ejemplo de la salida del preprocesamiento del sistema, si la entrada es:

```
El gato come pescado.
Pero a Don Jaime no le gustan los gatos.
```

Salida:

```
El el DA0MS0
gato gato NCMS000 01630731:07221232:01631653
come comer VMIP3S0 00794578:00793267
pescado pescado NCMS000 05810856:02006311
. . Fp
Pero pero CC
a a SPS00
Don_Jaime Don_Jaime NP00000
no no RN
le él PP3CSD00
gustan gustar VMIP3P0 01244897:01213391:01241953
los el DA0MP0
gatos gato NCMP000 01630731:07221232:01631653
. . Fp
```

Donde cada línea de la salida tiene el formato: palabra lemma categoría gramatical probabilidad de la categoría y sentidos de la palabra (synsets de WordNet 1.6).

El modulo desambiguador toma esta salida aplicándole en algoritmo propuesto, dando una salida de etiquetado a nivel de sentidos.

El sistema en su totalidad es una aplicación Web, llamado recursos visuales en la sección anterior, para en uso de un usuario final. Desarrollado en CGI (*Common Gateway Interface*) como se muestra en la Figura 9. Que permite a un cliente (explorador Web) solicitar datos de un programa ejecutado en un servidor Web. CGI especifica un estándar para transferir datos entre el cliente y el programa. Es un mecanismo de comunicación entre el servidor Web y una aplicación externa cuyo resultado final de la ejecución son objetos MIME. Las aplicaciones que se ejecutan en el servidor reciben el nombre de CGIs.

Las aplicaciones CGI fueron una de las primeras maneras prácticas de crear contenido dinámico para las páginas Web. En una aplicación CGI, el servidor Web pasa las solicitudes del cliente a un programa externo. Este programa puede estar hecho en cualquier lenguaje que soporte el servidor, aunque por razones de portabilidad se suelen usar lenguajes de script, siendo Perl la elección para la implementación. La salida de dicho programa es enviada al cliente en lugar del archivo estático tradicional.

CGI ha hecho posible la implementación de funciones nuevas y variadas en las páginas Web, de tal manera que esta interfaz rápidamente se volvió un estándar, siendo implementada en todo tipo de servidores Web.

Fig. 9 Aplicación Web

El usuario deberá introducir el texto a traducir en el cuadro de texto, después elegir el idioma fuente–destino y oprimir el botón de traducir, visto en la Figura 10. El servidor tomara la petición y la procesara, regresando el texto traducido, mostrado en la Figura 11.

Fig. 10 Uso de la aplicación Web

Fig. 11 Respuesta del servidor

El servidor es una maquina con 1 GB de RAM y 3.6 G Hz de procesador Pentium VI, la aplicaron Web se montó sobre un servidor HTTP Apache 2.2.

5.2.2. Módulos principales

Módulo Tokeniser

Divide el texto de entrada en *tokens* simples como números, signos de puntuación y palabras de diferentes tipos (por ejemplo distingue entre palabras en mayúsculas y en minúsculas).

- Tipos de Tokens:
- Word
- Number
- Punctuation

- Space Token

Módulo Gazetter

Este módulo utiliza una serie de listas de palabras, que se encuentran almacenadas en archivos de texto, con una entrada por línea. Cada lista representa un conjunto de nombres, por ejemplo nombres de ciudades, organizaciones, días de la semana.

Un archivo de texto funciona como índice (*lists.def*) y es utilizado para acceder a cada una de las listas en particular. Para cada lista se define un tipo principal y opcionalmente un subtipo. En el ejemplo a continuación la primera columna corresponde al nombre de la lista, la segunda al tipo principal y la tercera al subtipo.

- currency_prefix.lst:currency_unit:pre_amount
- currency_unit.lst:currency_unit:post_amount
- date.lst:date:specific
- day.lst:date:day

Estas listas son compiladas en máquinas de estado finito y cada *token* reconocido por alguna máquina es anotado con los valores del tipo principal y del subtipo (si está definido).

Módulo Sentence Splitter

Separa la oración distinguiendo sus límites. Es una cascada de transductores de estados finitos los cuales segmentan el texto en oraciones. Este módulo es requerido por el Tagger. El splitter utiliza el gazetter de abreviaturas para ayudarse a distinguir en fin de una oración.

Módulo Stemmer

Este modulo implementa el algoritmo de Porter-Stemmer. Reduce cada palabra a su raíz o stem (Ej. *blogging* a *blog*). Necesita anotación de Tokens.

Módulo Tagger

Brill Tagger, el cual produce PoS tags (etiquetas), como una anotación en cada palabra o símbolo. La lista de etiquetas usada esta dada en un archivo. El tagger utiliza un lexicon y un conjunto de reglas (resultado de un entrenamiento en un corpus).

Necesita de anotaciones del tipo Token y Sentence en el documento de entrada.

Módulo desambiguador

Encargado de etiquetar cada ocurrencia de la palabra ambigua con su sentido correcto. Su entrada es un documento preprocesado por los módulos anteriores en formato Senseval-2 y su salida, el documento etiquetado con sentidos. Mostrado su arquitectura en la Figura 12. se puede ver el flujo de datos por el modulo, así como los requerimientos de información del mismo.

El modulo es la implementación del algoritmo propuesto en este trabajo terminal, necesita de la interacción con WN para la fase de "encontrar sinónimos por cada palabra" utilizando el modulo de Perl, WordNet::QueryData (Rennie, 2000).

En la fase de "Realizar consulta" se pueden identificar dos escenarios, el primero, donde el numero de consultas es menor a 1000 (número permitido por Google de consultas al día), donde el modulo usará la dll de Windows (u otro modulo de Perl) Win32::Internet, para buscar directamente la información en Internet, y en un segundo escenario donde las consultas superan al máximo permitido la búsqueda se realizará en una base de datos local de Google. Para después utilizar las frecuencias recuperadas para el cálculo del sentido correcto en cada ocurrencia de la palabra ambigua.

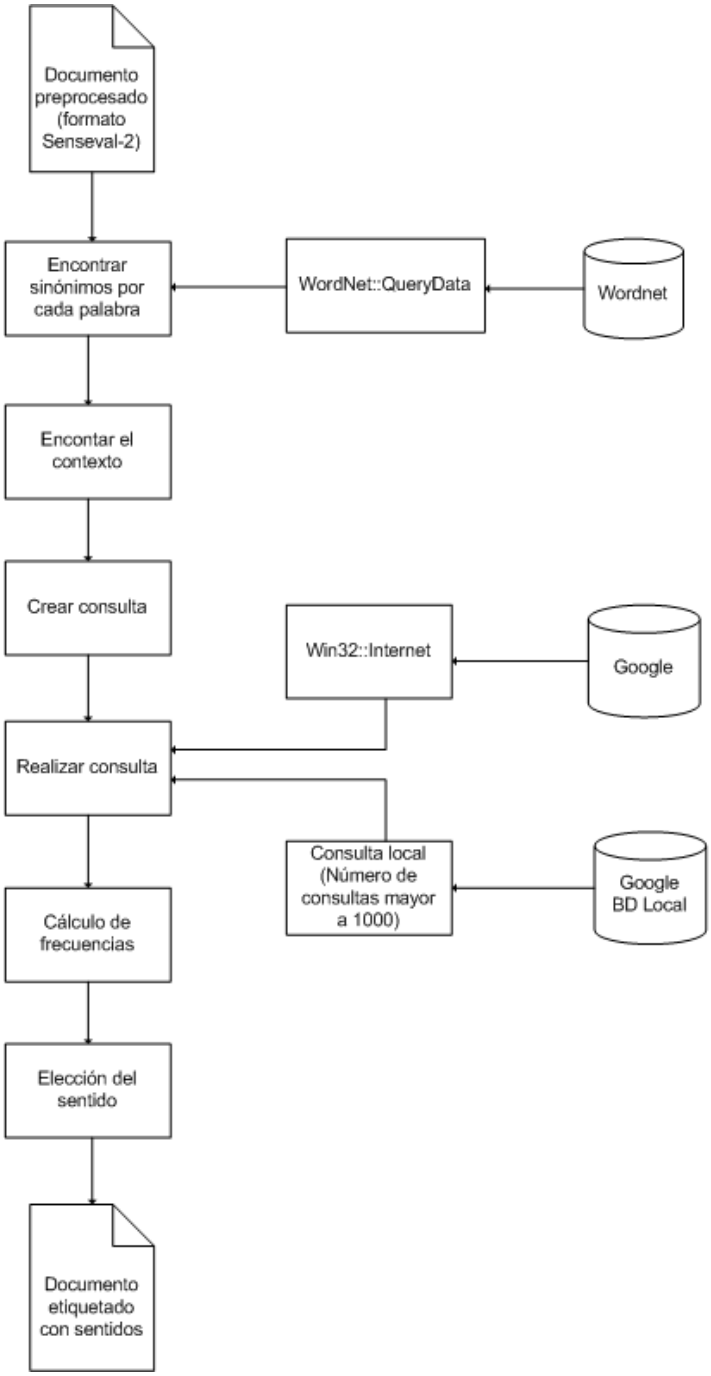

Fig. 12. Diagrama de bloques del modulo desambiguador

6. Resultados experimentales

Evaluar el desempeño de los algoritmos de WSD requiere de corpus anotados manualmente. Un corpus anotado manualmente es un conjunto de textos donde una persona experta ha asignado las etiquetas con sentidos usando un diccionario o una base de conocimiento como fuente de información de sentidos. Las etiquetas de sentidos asignadas por un programa son finalmente comparadas a las etiquetas asignadas manualmente.

Existen muy pocos corpus etiquetados manualmente. El más largo SemCor, el cual es un subconjunto del Brown Corpus. También existen corpus creados para Senseval-2 y Senseval-3.

Además los métodos de etiquetado manual no son muy fiables y consumen mucho tiempo en su realización. Los humanos normalmente están en desacuerdo con que sentido tiene la palabra en el texto. Las palabras suelen ser utilizadas de forma ambigua o a veces de forma incorrecta. Siendo un juicio de un humano diferente al de otro, por lo tanto no se espera que un algoritmo de WSD tenga un desempeño perfecto.

La tarea de *all–words* es mucho más difícil que la de *lexical sample task*. Donde la tarea de *lexical* tiene que desambiguar una palabra a través del texto. Por ejemplo, cada ocurrencia de la palabra *line* (línea) debe ser etiquetada en un corpus. Esto es mucho más sencillo para un humano experto, saber cada definición de una sola palabra y asignar etiquetas a cada instancia de esta que a todas las palabras de un corpus (*all–words*). Asignar una etiqueta a cada palabra del corpus requiere ver por cada una en diccionario. Así el etiquetado de todas las palabras puede ser menos preciso porque el experto no se vuelve muy familiar con cada definición de cada palabra.

6.1. Datos experimentales

El corpus de Brown fue creado por la universidad de Brown en 1964, Este corpus contiene una amplia selección de textos en ingles consistente en un millón de palabras. El corpus incluye artículos, ficción, religión y escritos científicos.

SemCor consiste de una selección de textos del corpus de Brown con alrededor de 360000 palabras de las cuales 221000 se encuentran etiquetadas con sentidos. Las palabras sin etiquetar no aparecen en WordNet. SemCor se encuentra separado en 186 archivos cada uno correspondiente con un archivo de Brown.

Desde que el corpus de Brown fue creado en 1964, este corpus no ha reflejado los cambios del lenguaje inglés, durante los pasados 40 años. Por otro lado WordNet, que

fue creado en 1990 es mejorado frecuentemente para reflejar los cambios del lenguaje.

Las palabras usadas en Semcor son etiquetadas usando WordNet 1.6. La versión utilizada en este trabajo terminal es la 2.1.

6.2. Metodología experimental

Estudios de (Resnik y Yarowsky, 1997) muestran la dificultad de comparar los métodos para la WSD, algunos problemas residen en distinciones como el enfoque (Basado en conocimiento, supervisado y no supervisado), como en las palabras a desambiguar. Los métodos propuestos fueron probados sobre 2 archivos etiquetados de SemCor (br-a01, br-a02). Utilizando WordNet como inventario de sentidos, y como fuente de sinónimos y glosas para lo dos métodos. Y Google como motor de búsqueda para las consultas.

6.3. Resultados

La Tabla 2 presenta la precisión obtenida por los dos métodos comparados con el *baseline*. Considerado el método más simple de WSD, en el cual cada palabra es etiquetada con el sentido más común, por ejemplo, el primer sentido definido en WordNet. El método de Satanjeev y Pedersen (2002) donde utilizan a WordNet como inventario de sentidos, llamado Lesk 1 en la Tabla 2. Y con el algoritmo de Lesk simplificado mostrado en (Mihalchea y Tarau, 2004), utilizando un *back–off* (escoger un sentido de manera aleatoria) llamado en la Tabla 2 Lesk 2.

Tabla 2. Comparación con el baseline

Semcor	baseline	Lesk 1	Lesk 2	Algoritmo 1	Algoritmo 2
sustantivos	80%	29%	47%	38%	34.5%

6.4. Comparación y discusión

Aunque el algoritmo 1 resulto por debajo del baseline y el algoritmo de Lesk 2, como también el algoritmo 2., este cuenta con la ventaja de no crear un problema combinatorio como el método de Lek original. Siendo este algoritmo, Lesk 1 superado por los dos métodos propuestos. Comparando los métodos propuestos entre si el algoritmo 2 tuvo una precisión menor que el algoritmo 1, pero este a su vez es mucho más rápido que el algoritmo 1, porque realiza menos consultas a la Web.

7. Conclusiones y trabajo futuro

En esta sección se muestran las conclusiones el trabajo futuro y una publicación relacionada con la tarea de WSD, pero abordando el problema desde otro punto de vista, el Enfoque lógico Combinatorio para el reconocimiento de patrones, ya que la WSD es en esencia un problema de clasificación, como se mencionó en el capitulo 1.

7.1. Conclusiones

Los métodos mostrados aunque no poseen mejores resultados, resuelven el problema combinatorio que el método de Lesk crea. Dondelos dos algoritmo tienen la ventaja de su simplicidad y al contrario de los métodos supervisados no depender de corpus etiquetados, los cuales son escasos y costosos de realizar. Mostrando que la Web puede ser utilizada como corpus y que el uso de palabras semánticamente cercanas puede dar un buen indicio para la solución de la WSD. Por su parte la WSD puede mejorar la tarea de traducción automática, con el problema de depender del diccionario bilingüe que se utilice.

7.2. Publicación preparada

En esta publicación se propone un método para la WSD basado en el algoritmo supervisado KORA-Ω. El cual tiene la ventaja de su simplicidad y el pequeño conjunto de rasgos que utiliza, a costo de su baja precisión a comparación de los complejos métodos del estado del arte. Esta publicación fue preparada para el Congreso Iberoamericano de Reconocimiento de Patrones (CIARP, 2008). La publicación se muestra en la sección final del anexo.

7.3. Trabajo futuro

Se planea experimentar en el algoritmo 1 con otro tipo de palabras no solo sinónimos, sino también hiperónimos u otras palabras semánticamente relacionadas. En lo que respecta a la medida de cohesión de las palabras utilizada se experimentara tanto con MI como con SCI, mostradas en el marco teórico.

El algoritmo 2 puede se modificado de la misma forma en lo respectivo a la medición de cohesión de las palabras, y el uso de ejemplos u otra información en la paso de buscar la coocurrencia de las palabras en la Web. También la consulta en le motor de busque da puede ser mejorada, para evitar el uso de otras medidas de cohesión

8. Referencias

Agirre, E. and Martinez, D (2001). Learning class-to-class selectional preferences. CONLL 2001.

Banerjee, S. and Pedersen, T (2002). An adapted Lesk algorithm for word sense disambiguation using WordNet, CICLING 2002.

Domingos and Pazzani (1997).On the Optimality of the Simple Bayesian Classifier under Zero-One Loss, Machine Learning (29).

Gale, W., Church, K., and Yarowsky, D (1992). Estimating upper and lower bounds on the performance of word-sense disambiguation programs, ACL 1992.

Gale, W., Church, K., and Yarowsky, D (1992). One sense per discourse, DARPA workshop

Iulia Nica (2006). El conocimiento lingüístico en la desambiguación semántica automática, Sociedad Española para el Procesamiento del Lenguaje Natural, Compobell, S.L.

Igor A. Bolshakov 1 and Elena I. Bolshakova (1999). Measurements of Lexico-Syntactic Cohesion by means of Internet,

J. Atserias, B. Casas, E. Comelles, M. González, L. Padró, M. Padró (2006).FreeLing 1.3: Syntactic and semantic services in an open-source PLN library,

Jim Cowie, Joe Guthrie, Louise Guthrie (1992). Lexical Disambiguation using Simulated Annealing, Proceedings of the 14th International Conference on Computacional Linguistics.

K. Bontcheva, V. Tablan, D. Maynard, H. Cunningham (2004). Evolving GATE to Meet New Challenges in Language Engineering. Natural Language Engineering. 10 (3/4), pp. 349-373.

Martinez, D. and Agirre, E (2000). One sense per collocation and genre/topic variations, EMPLN 2000.

Michael Lesk (1986). Automatic Sense Disambiguation Using Machine Readable Dictionaries: How to Tell a Pine Cone from an Ice Cream Cone, Proceedings of SIGDOC.

Miller, G (1995). Wordnet: A lexical database, ACM, 38(11) 1995.

Miller and Charles (1991). Contextual correlates of semantic similarity, Language and Cognitive Processes, 6.

Nancy Ide y Jean Véronis (1998). “Word Sense Disambiguation”, Computational Liguistics.

Patwardhan, S. and Banerjee, S. and Pedersen, T (2003). Using Measures of Semantic Relatedeness for Word Sense Disambiguation, CICLING 2003.

Purandare and Pedersen (2004). Word Sense Discrimination by Clustering Contexts in Vector and Similarity Spaces, Proceedings of the Conference on Natural Language and Learning.

Rada Mihalcea, Ted Pedersen (2004). Advances in Word Sense Disambiguation, Tutorial at IBERAMIA. LNCS

R. Mihalcea and I. Moldovan (1999). An Automatic Method for Generating Sense Tagged Corpora. Proc. of the 16th National Conference on Artificial Intelligence. AAAI Press.

Resnik, P (1995). Using information content to evaluate semantic similarity, IJCAI 1995.

Philip Resnik and David Yarosky (1997). Distinguishing systems and distinguishing senses: new evaluation methods for Word Sense Disambiguation, Natural Language Engineering, 5: 113-133 Cambridge University Press.

Rushton Prince (2005).Using RUP/UP: 10 Easy Steps, X-tier SAE Inc.

Rational Software White Paper (1998), Rational Unified Process Best Practices for Software Development Teams.

Sharam Hekmat (2005).UML Process, Pragsoft Corporation.

Yarowsky, D (1995). Unsupervised word sense disambiguation rivaling supervised methods, Proceedings of ACL 1995.

Yarowsky, D (1993). One sense per collocation, ARPA Workshop 1993.

A. Anexo

a) Ejemplos de datos de entrada

Este es el ejemplo de entrada del modulo desambiguador, utilizado para la experimentación, siendo este el formato de entrada de los dos algoritmos, sobre Semcor. Fragmento del archivo br-a02 .

<contextfile concordance=brown>

<context filename=br-a02 paras=yes>

<p pnum=1>

<s snum=1>

<wf cmd=done pos=NN lemma=committee wnsn=1 lexsn=1:14:00::>Committee</wf>

<wf cmd=done pos=NN lemma=approval wnsn=1 lexsn=1:04:02::>approval</wf>

<wf cmd=ignore pos=IN>of</wf>

<wf cmd=done rdf=person pos=NNP lemma=person wnsn=1 lexsn=1:03:00:: pn=person>Gov._Price_Daniel</wf>

<wf cmd=ignore pos=POS>'s</wf>

<punc>``</punc>

<wf cmd=done pos=JJ lemma=abandoned wnsn=1 lexsn=5:00:00:uninhabited:00>abandoned</wf>

<wf cmd=done pos=NN lemma=property wnsn=1 lexsn=1:21:00::>property</wf>

<punc>"</punc>

<wf cmd=done pos=NN lemma=act wnsn=1 lexsn=1:10:01::>act</wf>

<wf cmd=done pos=VB lemma=seem wnsn=1 lexsn=2:39:00::>seemed</wf>

<wf cmd=done pos=JJ lemma=certain wnsn=4 lexsn=3:00:03::>certain</wf>

<wf cmd=done pos=NN lemma=thursday wnsn=1 lexsn=1:28:00::>Thursday</wf>

<wf cmd=ignore pos=IN>despite</wf>

<wf cmd=ignore pos=DT>the</wf>

<wf cmd=done pos=JJ lemma=adamant wnsn=1 lexsn=5:00:00:inflexible:02>adamant</wf>

<wf cmd=done pos=NN lemma=protest wnsn=1 lexsn=1:10:00::>protests</wf>

<wf cmd=ignore pos=IN>of</wf>

<wf cmd=done pos=NN lemma=texas wnsn=1 lexsn=1:15:00::>Texas</wf>

<wf cmd=done pos=NN lemma=banker wnsn=1 lexsn=1:18:00::>bankers</wf>

<punc>.</punc>

</s>

</p>

Donde cada línea esta compuesta por una palabra y si esta tiene sentidos en WordNet. El formato, con etiquetas XML, de ella será: Dentro de la etiqueta <wf…> pos (categoría gramatical), lemma (raíz de la palabra), wnsn (etiquetado manual de WordNet), y fuera de la etiqueta la palabra ambigua.

En formato de entrada en el modulo desambiguador utilizado para el sistema de traducción es el de tipo de Freeling es:

The the DT 0.9998

FreeLing freeling NNP 1

package package NN 0.962199 03074890:05997364

consists consist VBZ 1 01810781

of of IN 0.999901

a 1 Z 0.99998

library library NN 1 02920211:02920454:02920588:05980872:05981110

providing provide VBG 1 00721124:00804159:01590833:01618684

language language NN 1
04370255:04475662:04748361:04759121:05287805:05319899

analysis analysis NN 1
00422134:00451738:04451818:04604475:04806277:05294454

services service NNS 0.498797
00062420:00172275:00205456:00371920:00379388:00379947:00384296:00547590:
00666638:00784280:03303697:04012006:06084772:06091176 services NNS
0.498797 00384420

((Fpa 1

such such JJ 0.915451 01048401:01048574:01491308

as as IN 0.834832

morfological morfological JJ 1

analysis analysis NN 1
00422134:00451738:04451818:04604475:04806277:05294454

, , Fc 1

date date NN 0.953333
05800663:06204296:07201979:10848812:10880047:10880228:10880601:10897608

recognition recognition NN 1 00130222:04445171:04474949:05014301:10341471

, , Fc 1

PoS pos NNP 1

tagging tag VBG 1 00697256:01094485:01102160:01167471:01365796

, , Fc 1

etc. etc. NN 1

)) Fpt 1

Donde cada línea contiene el formato de Freeling, descrito en el capitulo 5.

b) Ejemplos de formato de salida

El formato de salida el desambiguador utilizado en la experimentación con el algoritmo 1 es el siguiente:

<entrada palabra=Committee wnsn=1>

<lemma>committee</lemma>

<contexto>Committee approval property act</contexto>

<sentido=committee#n#1>

<sinonimo>committee#n#1</sinonimo>

<contexto>committee approval property act</contexto>

<sinonimo>commission#n#1</sinonimo>

<contexto>commission approval property act</contexto>

</sentido>

<sentido=committee#n#2>

<sinonimo>committee#n#2</sinonimo>

<contexto>committee approval property act</contexto>

<sinonimo>citizens_committee#n#1</sinonimo>

<contexto>citizens_committee approval property act</contexto>

</sentido>

<eleccion>1</eleccion>

<buena> 1

</entrada>

Al igual que Semcor la salida es al estilo de etiquetas de XML donde la etiqueta entrada contiene la palabra a desambiguar, después la etiqueta lemma contiene la raíz de la palabra, después viene el contexto en el que se encuentra. Siguiendo el proceso que toma el algoritmo 1, visto en un ejemplo del capitulo 4.

Pro su parte el desambiguador del sistema da una salida para el traductor al estilo Freeling como se muestra a continuación:

The

FreeLing

package 05997364

consists 01810781

of

a

library 02920211

providing 00721124

language 04759121

analysis 00451738

services 00547590

(

such 01048401

as

morfological

analysis 00451738

,

date 05800663

recognition 10341471

,

PoS

tagging 01365796

,

etc.

)

Donde cada línea tiene el formato de palabra y etiqueta de sentido dada por el algoritmo 1.

c) Código de los módulos principales

Los módulos principales son:

Traductor.cgi

```
#!c:/Perl/bin/perl.exe

use CGI qw/:standard/;

use RecursosDeProcesamiento;
```

```
    print header,
    start_html('Traductor 1'),
    h1('Texto a traducir'),
    start_form,
    "Texto original:",p,textarea('texto',",10,20),p,
    popup_menu(-name=>'lengpar',-values=>['inglés a español','español a inglés']),
    submit(-value=>'Traducir'),
    end_form,
    hr,"\n";
    if (param)
    {
    my $texto=param('texto');
    my $lengpar=param('lengpar');
    my $recurso=RecursosDeProcesamiento->new(texto=>$texto);
    $recurso->escribirArchivo;
    $recurso->setLengpar($lengpar);
    $recurso->procesar();
    $traduccion=$recurso->getTraduccion();
    print"Texto traducido de forma automática",p,$traduccion,p;
}
    print end_html;
```

RecursosDeProcesamiento.pm

```
package RecursosDeProcesamiento;

use recursosProcesamiento::Tokeniser;

use recursosProcesamiento::Gazetteer;

use recursosProcesamiento::SentenceSplitter;

use recursosProcesamiento::Stemmer;

use recursosProcesamiento::Tagger;

use recursosProcesamiento::Desambiguador;

use recursosProcesamiento::Traductor;

sub new

{

   my $clase=shift;

   my $recursosDeProcesamiento={@_};

   #atributos de la instancia

   $recursosDeProcesamiento->{texto}||=0;

   $recursosDeProcesamiento->{lengpar}||="0";

   $recursosDeProcesamiento->{archivo}||="texto$$.txt";

   $recursosDeProcesamiento->{traduccion}||="0";

   bless $recursosDeProcesamiento,$clase;

   return $recursosDeProcesamiento;

}
```

```
sub escribirArchivo

{

    my $recursosDeProcesamiento=shift;

    open(FILE,">$recursosDeProcesamiento->{archivo}")or die "no se puede abrir el archivo $archivo\n";

    print FILE "$recursosDeProcesamiento->{texto}";

    close(FILE);

}

sub leerArchivo

{

    my $recursosDeProcesamiento=shift;

    my $archivo=shift;

    $recursosDeProcesamiento->{archivo}=$archivo;

    open(FILE2,"$recursosDeProcesamiento->{archivo}")or die "no se puede abrir el archivo $archivo\n";

    my @traduccion=<FILE2>;

    $recursosDeProcesamiento->{traduccion}=join("\n",@traduccion);

    #print "T=@traduccion\n $recursosDeProcesamiento->{archivo}\n";

    close(FILE);

    return $recursosDeProcesamiento;

}

sub setLengpar
```

```
{
    my $recursosDeProcesamiento=shift;
    my $lengpar=shift;
    $recursosDeProcesamiento->{lengpar}=$lengpar;
    if($lengpar eq 'inglés a español')
    {
        $recursosDeProcesamiento->{lengpar}="1";
    }elsif($lengpar eq 'español a inglés')
    {
        $recursosDeProcesamiento->{lengpar}="2";
    }
return $recursosDeProcesamiento;
}
sub procesar
{
     my $recursosDeProcesamiento=shift;
     my $leng=$recursosDeProcesamiento->{lengpar};
    my $tokeniser=Tokeniser->new(leng=>$leng);
    #my $gazetteer=Gazetteer->new(leng=>$leng);
    my $sentenceSplitter=SentenceSplitter->new(leng=>$leng);
    my $stemmer=Stemmer->new(leng=>$leng);
```

```
    my $tagger=Tagger->new(leng=>$leng);
    my $desambiguador=Desambiguador->new(leng=>$leng);
    my $traductor=Traductor->new(leng=>$leng);
     my $archivo=$recursosDeProcesamiento->{archivo};
    $archivo=$tokeniser->llamarRecurso($archivo);
    #$archivo=$gazetteer->llamarRecurso($archivo);
    $archivo=$sentenceSplitter->llamarRecurso($archivo);
    $archivo=$stemmer->llamarRecurso($archivo);
    $archivo=$tagger->llamarRecurso($archivo);
    $archivo=$desambiguador->llamarRecurso($archivo);
    $archivo=$traductor->llamarRecurso($archivo);
    &leerArchivo($recursosDeProcesamiento,$archivo);
}
sub getTraduccion
{
    my $recursosDeProcesamiento=shift;
    return $recursosDeProcesamiento->{traduccion};

}
1;
```

Tokeniser.pm

```
package Tokeniser;

#use recursosProcesamiento::lecturaEscritura;

$espanol='C:\FreeLing\share\config\es.cfg';

$ingles='C:\FreeLing\share\config\en.cfg';

#Constructor

sub new

{

   my $clase=shift;

   my $tokeniser={@_};

   #atributos de la instancia

   $tokeniser->{archivo}||=0;

   $tokeniser->{archivoTokeniser}||="texto$$.tok";

   $tokeniser->{leng}||="0";

   bless $tokeniser,$clase;

   return $tokeniser;

}

sub llamarRecurso

{

   my $tokeniser=shift;

   my $archivo=shift;
```

```
    $tokeniser->{archivo}=$archivo;

    if($tokeniser->{leng} eq "2")

    {

      $exe="analyzer -f $espanol --outf token < $tokeniser->{archivo} > $to-
keniser->{archivoTokeniser}";

    }else

    {

      $exe="analyzer -f $ingles --outf token < $tokeniser->{archivo} > $tokeniser-
>{archivoTokeniser}";

    }

    $result=system $exe;

    return $tokeniser->{archivoTokeniser};

  }

  1;
```

d) Publicación enviada

Word Sense Disambiguation with the KORA- Algorithm

Abstract. We present a method for word sense disambiguation (WSD) basing on the KORA- supervised learning algorithm. The advantage of the method is its simplicity and a very small feature set used, though, as we show, this is achieved at the cost of lower accuracy of the final result than the complex state-of-the-art methods achieve.

1 Introduction

Word Sense Disambiguation (WSD) is the processes of selecting the most appropriate meaning for a word basing on the context in which it occurs. For example, in the phrase *The bank down the street was robbed*, the word *bank* means a financial institution, while in *The city is on the Western bank of Jordan*, this word refers to the shore of a river. The WSD is an intermediate task [17] in natural language processing chain, essential for applications such as information retrieval or machine translation.

It can be thought of as a classification task, where word senses are the classes, the context provides the evidence, and each occurrence of a word is assigned to one of its possible classes basing on the evidence. This task is often treated as a supervised learning problem, where a classifier is trained from a corpus of manually sense-tagged texts using machine learning methods. These approaches typically represent the context in which each sense-tagged instance of the ambiguous word occurs using the features such as the part-of-speech (PoS) of surrounding words, keywords, syntactic relationships, etc.

To address this task, different statistical methods have been proposed, with various degrees of success. This includes a number of different classifiers like Naïve Bayes [12], neural networks [5], and content vector-based classifiers [8].

In this paper we present a supervised learning method based on the Logical Combinatorial Pattern Recognition (LCPR) approach, called KORA- [4].

The paper begins with a review of the related works in the area. In Section 3 the proposed method is presented. Section 4 reports our experimental results. Section 5 concludes the paper and introduces some future work.

2 Related Work

Several research projects take a supervised learning approach to WSD [3, 6, 8]. The goal is to learn to use surrounding context to determine the sense of an ambiguous word.

Often the disambiguation accuracy is strongly affected by the size of the corpus used in the process. Typically, 1000–2500 occurrences of each word are manually tagged in order to create a corpus. From this about 75% of the occurrences are use for the training phase and the remaining 25% are use for the testing [11]. Corpus like interest and line were the most well studied in literature.

The *Interest* dataset (a corpus where each occurrence of the word interest is manually marked up with one of its 6 senses) was included in a study by [13], who represent the context of an ambiguous word with the part-of-speech of three words to the left and right of interest, a morphological feature indicating if interest is singular or plural, an unordered set of frequently occurring keywords that surround interest, local collocations that include interest, and verb-object syntactic relationships. A nearest-neighbor classifier was employed and achieved an accuracy of 87% over repeated trials using randomly training and test sets. Ng and Lee [7], and Pedersen et al. [15] present studies that utilize the original Bruce and Wiebe feature set and include the interest data .The first compares a range of probabilistic model selection methodologies

and finds that none out perform the Naive Bayesian classifier, which attains accuracy of 74%. The second compares a range of machine learning algorithms and finds that a decision tree learner 78% and a Naïve Bayesian classifier 74% are most accurate.

The *Line* dataset (similarly, a corpus where each occurrence of the word *line* is marked with one of its 6 senses) was first studied by Leacock [8]. They evaluate the disambiguation accuracy of a Naive Bayesian classifier, a content vector, and a neural network. The context of an ambiguous word is represented by a bag-of-words (BoW) where the window of context is two sentences wide. When the Naive Bayesian classifier is evaluated words are not stemmed and capitalization remains. With the content vector and the neural network words are stemmed and words from a stop-list are removed. They report no significant differences in accuracy among the three approaches; the Naïve Bayesian classifier achieved 71% accuracy, the content vector 72%, and the neural network 76%.

This dataset was studied again by Mooney [12], where seven different machine learning methodologies are compared. All learning algorithms represent the context of an ambiguous word using the BoW with a two sentence window of context. In these experiments words from a stop list are removed, capitalization is ignored, and words are stemmed. The two most accurate methods in this study proved to be a Naive Bayesian classifier 72% and a perceptron 71%.

Recently, the Line dataset was revisited by both Towell and Voorhees [5], and Pedersen [14]. Take an ensemble approach where the output from two neural networks is combined; one network is based on a representation of local context while the other represents topical context. The latter utilize a Naive Bayesian classifier. In both cases context is represented by a set of topical and local features. The topical features correspond to the open-class words that occur in a two sentence window of context. The local features occur within a window of context three words to the left and right of the ambiguous word and include co-occurrence features as well as the PoS of words in this window. These features are represented as local and topical BoW and PoS. [5] report accuracy of 87% while [15] report accuracy of 84%.

3 Proposed Method

The KORA- algorithm is an extension of the widely used KORA-3 [2] in geosciences. This algorithm was used for supervised classification problems. The algorithm works with disjointed classes and objects. The idea is to classify new objects (patterns) based on a training sample of objects through the verification of some complex properties. These complex properties are a combination of certain feature values, named complex features (CF) that discriminate an object or a set of objects in the same class from the remaining objects in different classes. Fuzzy KORA- allows us to solve supervised classification problems with many classes (hard-disjointed or fuzzy), with any kind of features. In this model, complex properties could be of any length greater or equal to one.

The algorithm works based on the idea of finding for each object of the training matrix a property such there is a few other identical property in any object of the remaining classes. In general, we can describe this family of algorithms in three stages:

Learning stage It is necessary all the parameters in order to determine which property of feature values are complex features for each class. The entire objects in each class are covered for enough complex features. Enough is also a parameter of the algorithm. All the classified objects in each class satisfy at least a predetermined number of complex features**.**

Relearning stage In this second stage the problem is the same as in the previous one, but with other parameters, shorter than the "enough" of the previous stage**.**

Classification stage Finally, we have all the complex features for each class, with different levels of discrimination (from the learning stage and the relearning stage). If a given complex feature for a class is present in the new object to classify, this class receives a vote. After that, any decision-making rule is applied**.**

3.1 Algorithm

The disambiguation of a particular word W is performed as follows:

INPUT: semantically untagged pattern of W and its context.
OUTPUT: semantically tagged pattern of W and its context.

Learning Stage

Step 1. Define β_1, and β_2 thresholds.
Step 2. Define the properties.
Step 3. Find the characteristic CF, β_1-characteristic.

Relearning stage

Step 4. Calculate the rest of the class, using the definition of complement.
Step 5. Define β_3, and β_4 thresholds
Step 6. Find the complementary CF, β_3-complementary

Classification Stage

Step 7. Apply a decision-making rule.

3.2 Decision-making Rule

The final step of the classification stage is divided in 3 sub-steps:

Calculate the voting scheme. If an object fulfill with an a characteristic CF, then it gives one vote. For a complementary CF the process is the same. After a counting of the CF's we weight the characteristics voting with 0.7 and the complementary with 0.3. Finally the vote of the class is the sum of both characteristic and complementary CF**.**

Class membership. A pattern is assigned to the class with the biggest sum of CF**.**

Amount of membership. The pattern is assigned with membership degree 1 to the class of the previous step and 0 to the others**.**

4 Experimental Results

4.1 Data Set

The Line dataset was developed for the task of disambiguation of the word *line* into one of six possible senses (text, formation, division, phone, cord, and product) based on the words occurring in the current and previous sentence. The corpus was assembled from the 1987-89 Wall Street Journal and 25 million word corpus from the American Printing House for the Blind. Sentence containing *line* were extracted and assigned a single sense from WordNet [9]. There are a total of 4,149 examples in the full corpus unequally distributed across the six senses. This dataset and distribution of senses are shown in Table 1.

In this work, we used a subset of the Line dataset in which every sense is equally distributed taking 349 sense-tagged examples for each sense resulting in a training corpus of 2094 sense-tagged sentences. We form every sentence in a pattern format using only 3 open-class words to the left and right around the ambiguous word and leaving only the word an each PoS. We use for tuning the thresholds $\beta_1 = 3$, $\beta_2 = 0$, $\beta_3 = 2$, and $\beta_4 = 0$, see Section 3.

Table 3. Distribution of senses in Line dataset.

Sense	count
Product	2218
written or spoken text	404
telephone connection	429
formation of people or things; queue	349
an artificial division; boundary	376
a thin, flexible object; cord	373
Total:	4149

4.2 Complex Features

The local context is a window of lexical units that occur around the ambiguous word, varies from few words to the entire sentence. Some parameters that have been used are: distance, collocation, and syntactic information.

The concept of distance is related with the number of words (n) in the context. Studies gave different answers for an optimal number of n, Ide and Veronis [10], have shown that 2 words are enough for the WSD task, even 1 word is trustful. Other studies [18], reach the conclusion of an optimal n value for a local context in 3 or 4. Using in this paper as feature set of 3 words around the ambiguous word to the right and left. Creating 9 complex features sets, where are the combinations of words, and other sets for the PoS. For example the set CF_1={P:0,P:+1}, where P is the word and the number is the position in context.

4.3 Comparison with Previous Results

The best result of our method was achieved by using the complex features of words with a decreasing of accuracy using only the PoS complex features. Table 2 shows the accuracy compared to other methods, as evaluated in the Line dataset.

Table 4. Comparasion with previous results.

Method	Accuracy	Algorithm	Feature set
Pedersen 2000	88%	Naive Bayesian Ensemble	varying left & right; BoW
Towell & Voorhess 1998	87%	Neural network	local & topical BoW; PoS
Leacock, Chodorow & Miller 1998	84%	Naive bayes	local & topical BoW; PoS
Leacock Towell & Voorhees 1993	76% 72% 71%	Neural network Content vector Naive bayes	2 sentence BoW
Mooney 1996	72% 71%	Naive bayes Perceptron	2 sentence BoW
Proposed	60% 53%	KORA- KORA-	3 Word properties 3 PoS properties

Our algorithm uses a very limited feature set, even though at the cost of lower results as compared to complex state-of-the-art techniques. We believe that the algorithm could give better results by using more information of the context—for example, a wider window for both the words and PoSs or the use of information other than lexical, e.g., morphological. However, with this our algorithm would possibly lose its main advantage: simplicity.

Another possible improvement for the method is selecting of less restrictive complex features and thresholds such as β_2 and β_4, to permit any repetition of a complex feature in the other classes—things that KORA-3 does not allow.

5 Conclusions and Future Work

In this paper we used KORA- algorithm for the WSD task. This algorithm has the advantage of simplicity and the use of a very limited feature set, though at the cost of the accuracy of the final result.

Essentially, the KORA- makes a positive characterization of a class (properties which belongs to the class) basing on the idea of majorities and minorities of the population. For future work we will try to also make negative characterization of a class: properties which no belong to the class, as the Representative Set algorithms do [1]. We also plan to experiment with adding a limited subset of linguistically-motivated features, as well to try wider windows and more open thresholds.

References

1. Baskakova, L.V., and Yu.I. Zhuravlev. 1981. Model of algorithm of recognition with sets and support sets systems. Journal Zhurnal Vichislitielnoi Matemati y Matematicheskoi Fisiki 21, No. 5, 1264.
2. Bongard, M. N. et al. 1963. Solving geological problems using recognition programs. Journal Soviet Geology, 6C. Leacock, M. Chodorow, and G. Miller. 1998. Using corpus statistics and WordNet relations for sense identification. Computational Linguistics,24(1):147–165, March.
3. Brown, P., Della-Pietra, S., Della-Pietra, V., & Mercer, R. 1991. Word sense disambiguation using statistical methods. In Proceedings of the 29th Annual Meeting of the Association for Computational Linguistics, pp. 264–270.
4. De-la-Vega-Doria, L., J.A., Ruiz-Shulcloper, J., and Carrasco-Ochoa. 1998. Fuzzy KORA- algorithm. Proceeding of the Sixth European Congress on Intelligent Techniques and Soft Computing, EFIT'98, 1190–1194. Aachen, Germany
5. G.Towell and E.Voorhees.1998.Disambiguating highly ambiguous words. Computational Linguistics, 24(1):125–146, March.
6. Gale, W., Church, K., & Yarowsky, D. 1992. A method for disambiguating word senses in a large corpus. Computers and the Humanities,26, 415–439.
7. H.T. Ng and H.B. Lee. 1996. Integrating multiple knowledge sources to disambiguate word sense :An exemplar-based approach. In Proceedings of the 34th Annual Meeting of The Society for Computational Linguistics, pages 40–47.
8. Leacock, C., Towell, G., & Voorhees, E. 1993.Corpus-based statistical sense resolution. In Proceedings of the ARPA Workshop on Human Language Technology.
9. Miller, G. 1991. WordNet: An on-line lexical database. International Journal of Lexicography, 3(4).
10. Nancy Ide and Jean Véronis.1998.Word sense disambiguation: The state of the art. Computational Linguistics.
11. Rada Mihalcea and Dan Moldovan. 1999. An Automatic Method for Generating Sense Tagged Corpora, in Proceedings of the American Association for Artificial Intelligence, Orlando, FL, July.
12. R. Mooney. 1996. Comparative experiments on disambiguating word senses: An illustration of the role of bias in machine learning. In Proceedings of the Conference on Empirical Methods in Natural Language Processing, pages 82–91.
13. R. Bruce and J. Wiebe. 1994. Word-sense disambiguation using decomposable models. In Proceedings of the 32nd Annual Meeting of the Association for Computational Linguistics, pages 139–146.
14. Ted Pedersen. 2000. A simple approach to building ensembles of Naive Bayesian classifiers for word sense disambiguation. Proceedings of the first conference on North American chapter of the Association for Computational Linguistics. Seattle, Washington, pages 63–69.
15. T. Pedersen, R. Bruce, and J. Wiebe. 1997. Sequential model selection for word sense disambiguation. In Proceedings of the Fifth Conference on Applied Natural Language Processing ,pages 388–395,Washington, DC, April.

16. T. Pedersen and R. Bruce. 1997. A new supervised learning algorithm for word sense disambiguation. In Proceedings of the Fourteenth National Conference on Artificial Intelligence, pages 604–609.
17. Wilks, Yorick and Stevenson, Mark . 1996. The grammar of sense: Is word sense tagging much more than part-of speech tagging? .Technical Report CS-96-05, University of Sheffield, Sheffield, United Kingdom.
18. Yarowsky, David. 1994. "Decision Lists for Lexical Ambiguity Resolution: Application to Accent Restoration in Spanish and French," in Proceedings of the 32nd Annual Meeting of the Association .for Computational Linguistics, Las Cruces, NM.