Estudio sobre métodos tipo Lesk usados para la desambiguación de sentidos de palabras

Sulema Torres-Ramos

Centro de Investigación en Computación (CIC-IPN),
Unidad Profesional Adolfo López Mateos,
Av. Juan de Dios Bátiz s/n esquina M. Othón de Mendizábal,
Zacatenco, México, D.F. 07738, México.
sulema7@gmail.com

Resumen. La ambigüedad semántica es un problema que se presenta en todos los lenguajes naturales. Podríamos decir que para los seres humanos la ambigüedad en el lenguaje pasa desapercibida, debido a que la resolvemos casi inconscientemente utilizando la realidad en que vivimos, el contexto y el conocimiento que poseemos sobre algunos temas. Pero para las computadoras no es así. En el área de procesamiento de lenguaje natural, la tarea de desambiguación de sentidos de palabras es el problema de seleccionar un sentido, de un conjunto de posibilidades predefinidas, para una palabra dada en un texto o discurso. La desambiguación del sentido de las palabras, es considerada como uno de los problemas más importantes de investigación en el procesamiento del lenguaje natural. Es esencial para las aplicaciones que requieren la comprensión del lenguaje, como la comunicación hombre-máquina, traducción automática, recuperación de la información y otros. Uno de los primeros métodos propuestos para llevar a cabo esta tarea es el método de Lesk, el cual propone utilizar la coherencia global del texto, es decir, el total de sentidos de palabras relacionadas en el texto. La ventaja de este método es que sólo se necesita un diccionario de sentidos como recurso léxico. El problema principal es que mientras más palabras tengamos, más grande es el espacio de búsqueda. Por lo tanto, se han desarrollado diferentes métodos (conocidos como métodos tipo Lesk) que aplican modificaciones de este algoritmo intentando obtener la combinación de sentidos óptima para un texto dado. En este artículo se presenta un estudio de los principales métodos tipo Lesk usados para la desambiguación de sentidos de palabras.

Palabras clave: Lingüística Computacional, Procesamiento de Lenguaje Natural, Desambiguación de Sentidos de Palabras, Algoritmo de Lesk.

1 Introducción

La información es el recurso más importante que poseemos los seres humanos. Gran parte de esta información se comunica, almacena y maneja en forma de lenguaje natural (español, inglés, ruso, etc.). En la actualidad, podemos obtener grandes volúmenes de información en forma escrita, ya sea de manera impresa o electrónica.

Las computadoras son una herramienta indispensable para el procesamiento de la información plasmada en los textos, ya que son capaces de manejar grandes volúmenes de datos. Sin embargo, una computadora no puede hacer todo lo que las personas normalmente hacemos con el texto, por ejemplo, responder preguntas basándose en la información proporcionada, o, hacer inferencias lógicas sobre su contenido, o elaborar un resumen de esta información.

Por lo anterior, el Procesamiento de Lenguaje Natural (PLN) tiene por objetivo habilitar a las computadoras para que entiendan el texto, procesándolo por su sentido.

Para llevar a cabo esta tarea, un sistema de PLN necesita conocer sobre la estructura del lenguaje, la cual se analiza normalmente en los siguientes niveles [1]:

- Nivel fonológico: trata de los sonidos que componen el habla, permitiendo formar y distinguir palabras.
- Nivel morfológico: trata sobre la estructura de las palabras y las leyes para formar nuevas palabras a partir de unidades de significado más pequeñas llamadas morfemas.
- Nivel sintáctico: trata sobre cómo las palabras pueden unirse para construir oraciones y cuál es la función que cada palabra realiza en esa oración.
- Nivel semántico: trata del significado de las palabras y de cómo se unen para dar significado a una oración.
- Nivel pragmático: estudia la intención del hablante al producir oraciones específicas o textos en una situación específica.

Todos los niveles anteriores de la estructura del lenguaje tienen un problema: la ambigüedad.

La ambigüedad, en el proceso lingüístico, se presenta cuando pueden admitirse distintas interpretaciones a partir de una representación dada o cuando existe confusión al tener diversas estructuras y no tener los elementos necesarios para eliminar las eventualmente incorrectas. Para desambiguar, es decir, para seleccionar los significados o las estructuras más adecuadas de un conjunto conocido de posibilidades, se requieren diversas estrategias de solución según el caso [2].

Debido a que existe ambigüedad aún para los humanos, su solución no es sólo lograr la asignación del sentido único por palabra en el análisis de textos, sino eliminar la gran cantidad de variantes que normalmente existen. La ambigüedad es el problema más importante en el procesamiento de textos en lenguaje natural, por lo que su resolución es la tarea más importante a llevar a cabo.

Se distinguen tres tipos principales de ambigüedad: léxica, sintáctica (estructural) y semántica.

La ambigüedad léxica se presenta cuando las palabras pueden pertenecer a diferentes categorías gramaticales, por ejemplo, la palabra *bajo* puede ser una preposición, un sustantivo, un adjetivo o una conjugación del verbo bajar [3].

La ambigüedad sintáctica, también conocida como ambigüedad estructural se presenta cuando una oración puede tener más de una estructura sintáctica. Por ejemplo de la oración "*María habló con el profesor del instituto*" se puede entender dos cosas diferentes: a) el profesor pertenece al instituto, o bien, b) el tema del que habló María con el profesor fue el instituto [4].

La ambigüedad semántica se presenta cuando las palabras tienen múltiples significados, por ejemplo la palabra *banco* puede significar institución financiera, la orilla del lago, asiento, etc.

Hoy en día, cualquier palabra que usamos para comunicarnos tiene dos o más posibles interpretaciones, llamadas sentidos. Para entender correctamente un texto, el lector –humano o programa de computadora– debe ser capaz de determinar el sentido adecuado para cada palabra en el texto.

Además de entender un texto, hay muchas aplicaciones de procesamiento de lenguaje natural donde la determinación automática del sentido correcto de una palabra es crucial. Entre ellas se encuentran:

1. Traducción automática. La desambiguación semántica es esencial para la traducción apropiada de palabras como *bank*(banco) que, dependiendo del contexto, puede traducirse como *institución bancaria*, *orilla del río*, etc. [5,6].
2. Recuperación de información. Al realizar búsquedas por palabras clave específicas, es necesario eliminar los documentos donde se usa la palabra o palabras en un sentido diferente al deseado; por ejemplo, al buscar referencias sobre animales con la palabra *gato*, es deseable eliminar los documentos que contienen dicha palabra asociada con mecánica automotriz [7,8,9,10,11,20,21].
3. El procesamiento de texto. La desambiguación es necesaria para algunas tareas de procesamiento de texto, por ejemplo, para determinar cuándo deben insertarse acentos diacríticos [12,13] y para la detección y corrección del malapropismo [14,15,16].
4. Respuesta automática a preguntas (QA, por sus siglas en inglés: Question Answering): La meta de esta tarea es encontrar respuestas en el dominio de texto en lenguaje natural [17,18]. A diferencia de un sistema de recuperación de información que te devuelve los documentos relativos a un criterio de búsqueda, un sistema de QA devolverá una respuesta específica a la búsqueda especificada. Ejemplo: si buscamos "Patente bulbo Edison" un sistema de recuperación de información devolverá la lista de documentos relevantes que tengan que ver con la patente de bulbos de Edison, sin embargo un sistema de QA preguntará "¿Cuándo se registró la patente de bulbo de Edison?" y el sistema devolverá una respuesta específica.

En el área de procesamiento de lenguaje natural o lingüística computacional, la identificación del sentido de palabras en un contexto dado es conocida como desambiguación de sentidos de palabras (WSD por sus siglas en inglés).

Una forma de llevar a cabo la desambiguación de sentidos de palabras es tomar en cuenta la coherencia global del texto [19], es decir, el total de sentidos de palabras relacionadas en el texto. La limitación principal de esta técnica es que, para encontrar la combinación óptima de sentidos se necesita mucho tiempo, ya que el espacio de búsqueda es muy grande. La principal ventaja de esta técnica es que es un método no supervisado y sólo utiliza un diccionario de sentidos como recurso externo.

Debido a lo anterior, existen diferentes métodos (conocidos como métodos tipo Lesk) que aplican modificaciones del algoritmo original de Lesk para desambiguar palabras en un texto.

En este artículo se presenta una revisión de los principales métodos tipo Lesk usados para llevar a cabo la desambiguación de sentidos de palabras como una tarea del procesamiento de lenguaje natural.

El artículo se organiza como sigue: primero, formalizamos la tarea de desambiguación de sentidos de palabras y los métodos usados para esta tarea clasificados de acuerdo a los recursos que utilizan (sección 2). Después, se explica a detalle el algoritmo original de Lesk (sección 3) y se presenta un estudio sobre los principales métodos que se basan en dicho algoritmo (sección 4). En la sección 5 se presenta un análisis de los resultados obtenidos para los métodos tipo Lesk y al final se presentan las conclusiones.

2 Desambiguación del sentido de las palabras (WSD)

En general, la desambiguación del sentido de las palabras es el problema de seleccionar un sentido de un conjunto de posibilidades predefinidas para una palabra dada en un texto o discurso.

En los últimos años se han incrementado las investigaciones para crear métodos de WSD. A continuación se describe la clasificación para métodos de WSD de acuerdo a los recursos que utilizan (figura 1).

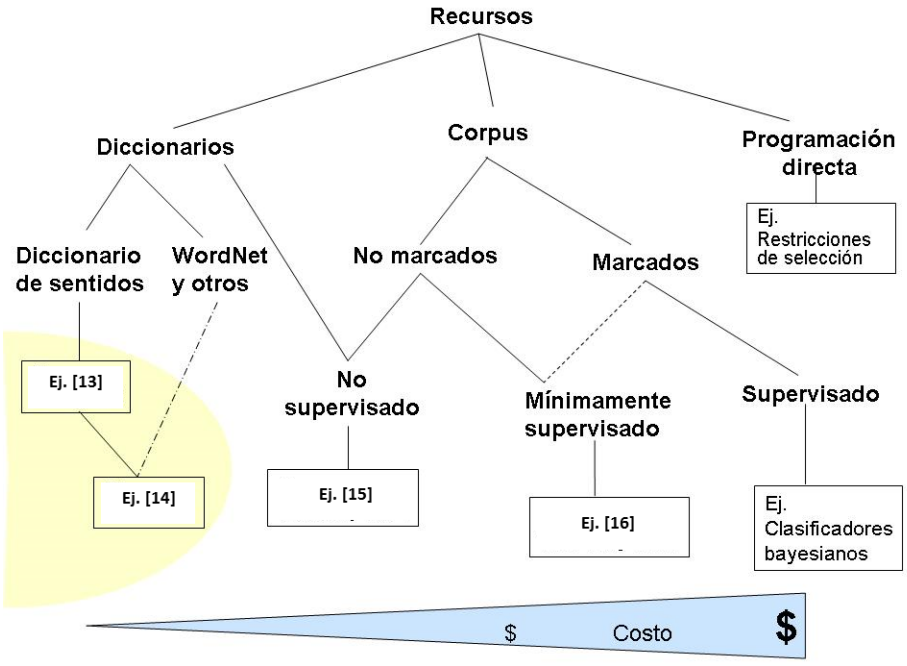

Fig. 1. Clasificación de los métodos para WSD de acuerdo a los recursos que utilizan.

2.1 Clasificación de métodos para desambiguación de sentidos de palabras

Los métodos para desambiguación de sentidos de palabras se clasifican en: los que utilizan diccionarios, los que utilizan corpus, y los que no utilizan ningún recurso léxico.

Los que utilizan ***diccionarios***:

Los diccionarios pueden ser *de sentidos* y otros como *WordNet*.

Los diccionarios proporcionan una lista de glosas (definición de sentido) para las palabras. Los métodos que utilizan sólo diccionarios de sentidos, buscan elegir un sentido (de esta lista) para cada palabra en un texto dado, tomando en cuenta el contexto en el que aparece. Como ejemplo, [19] propone utilizar la coherencia global del texto, es decir, el total de sentidos de palabras relacionadas en el texto: mientras más relacionadas estén las palabras entre sí, más coherente será el texto.

Además existen variantes del algoritmo de Lesk que utilizan no sólo diccionarios de sentidos, sino también otro tipo de diccionarios como WordNet.

Los que utilizan ***corpus***:

Los corpus pueden ser *no marcados* y *marcados*.

Los métodos que utilizan corpus no marcados son los no supervisados, estos métodos también utilizan otros recursos como WordNet para poder asignar un sentido a cada palabra que aparece en los textos no marcados. Como ejemplo de éstos tenemos el método de [22], el cual elige de un diccionario (tesauro) las palabras relacionadas con la palabra a desambiguar. Cada palabra relacionada tiene un peso, éstas y la palabra a desambiguar tienen sentidos en un diccionario. Para elegir el sentido correcto, las palabras relacionadas votan por un sentido de la palabra a desambiguar con cierto peso. Se elige el sentido con más peso.

Los métodos que utilizan corpus marcados son los métodos supervisados. Éstos reducen la desambiguación de sentidos de palabras a un problema de clasificación, donde a una palabra dada se le asigna el sentido más apropiado de acuerdo a un conjunto de posibilidades, basadas en el contexto en el que ocurre. Hay muchos algoritmos de aprendizaje supervisado utilizados para WSD, como ejemplo tenemos los clasificadores bayesianos, máquinas de soporte vectorial, árboles y listas de decisión, etc. [23].

Hay métodos que utilizan una gran cantidad de corpus no marcados [24] y muy pocos marcados [25] llamados mínimamente supervisados. Como ejemplo de éstos tenemos el método de [26], el cual identifica todas las ocurrencias de una palabra a desambiguar en un corpus no marcado. Después identifica un número pequeño de colocaciones semilla representativos de cada sentido de la palabra y etiqueta todos los ejemplos que contienen la colocación semilla con la palabra de dicha colocación (así tenemos los conjuntos etiquetados con cada sentido representativo y el conjunto residuo). El algoritmo utiliza los conjuntos etiquetados para entrenar una lista de decisión y encontrar nuevas colocaciones, para después etiquetar sobre el conjunto residuo. El algoritmo termina cuando el conjunto residuo se estabiliza.

Los que utilizan ***programación directa***:

Estos métodos se basan en reglas (muchas) que especifican el sentido de una palabra de acuerdo al contexto en el que aparece. Un ejemplo son las restricciones de selección (selectional restrictions), definen reglas de acuerdo a la palabra a desambiguar y su argumento. Ejemplo: el verbo *comer* puede tener como restricción que su tema argumento sea comida (comer-comida).

Este artículo se enfoca en los métodos que se basan en la ***aplicación directa de diccionarios de sentidos***, que se describen a continuación

3 Algoritmo de Lesk

El algoritmo de Lesk [19] es uno de los primeros algoritmos exitosos usados en la desambiguación de sentidos de palabras. Este algoritmo se basa en dos puntos principales: un algoritmo de optimización para WSD y una medida de similitud para las definiciones de los sentidos.

El primer punto es acerca de desambiguar palabras considerando la coherencia global del texto, esto es, encontrar la combinación de los sentidos que maximice la relación total entre los sentidos de todas las palabras.

Por ejemplo, para la oración *My father deposits his money in a bank account* y considerando a lo más tres sentidos[1] (véase tabla 1), para cada palabra, la figura 2 muestra la representación gráfica del algoritmo original de Lesk.

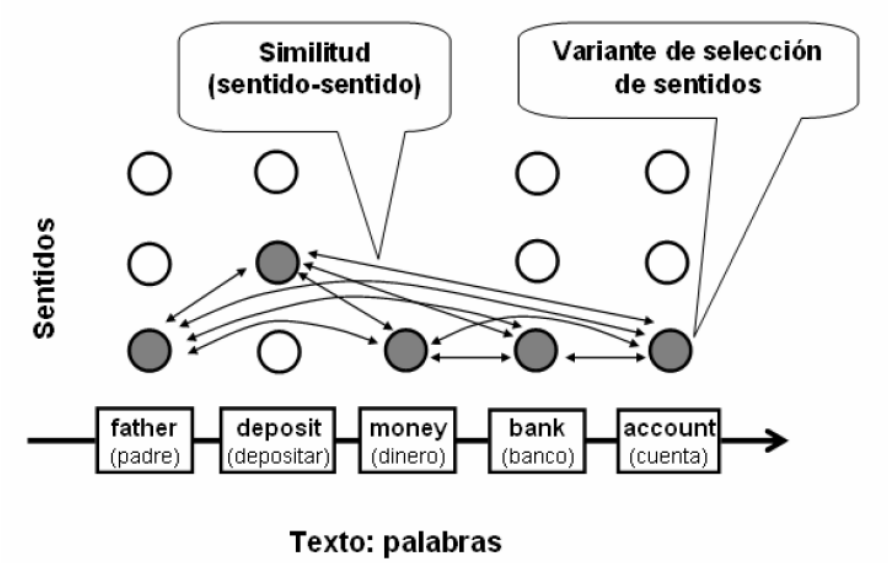

Fig. 2. Representación gráfica del algoritmo original de Lesk.

[1] Sentidos obtenidos de WordNet

Tabla 1. Sentidos de las palabras (máximo tres) obtenidas de WordNet para la oración "*My father deposits his money in a bank account*".

Palabra	Sentidos
Father	1: a male parent (also used as a term of address to your father); "his father was born in Atlanta". 2: `Father' is a term of address for priests in some churches (especially the Roman Catholic Church or the Orthodox Catholic Church); "`Padre' is frequently used in the military". 3: a person who holds an important or distinguished position in some organization; "the tennis fathers ruled in her favor"; "the city fathers endorsed the proposal".
Deposit	1: fix, force, or implant; "lodge a bullet in the table". 2: put into a bank account; "She deposits her paycheck every month". 3: put (something somewhere) firmly; "She posited her hand on his shoulder"; "deposit the suitcase on the bench"; "fix your eyes on this spot".
Money	1: the official currency issued by a government or national bank; "he changed his money into francs".
Bank	1: a financial institution that accepts deposits and channels the money into lending activities; "he cashed a check at the bank"; "that bank holds the mortgage on my home". 2: sloping land (especially the slope beside a body of water); "they pulled the canoe up on the bank"; "he sat on the bank of the river and watched the currents". 3: a supply or stock held in reserve for future use (especially in emergencies)
Account	1: a formal contractual relationship established to provide for regular banking or brokerage or business services; "he asked to see the executive who handled his account". 2: the act of informing by verbal report; "he heard reports that they were causing trouble"; "by all accounts they were a happy couple". 3: a record or narrative description of past events; "a history of France"; "he gave an inaccurate account of the plot to kill the president"; "the story of exposure to lead".

En el segundo punto, relacionado con la medida de similitud, Lesk sugiere usar el *traslape* entre las definiciones de los sentidos, es decir, contar el número de palabras que tienen en común.

Como ejemplo, para la oración, "*My father deposits his money in the bank account*" para medir la relación de las definiciones de los sentidos para la palabra "*deposit*" y "*bank*" como Lesk lo propuso, es necesario contar las palabras en común en todas las definiciones. En este caso, comparando principalmente las tres definiciones de "*deposit*" contra las tres definiciones de "*bank*". La relación entre los valores se muestra en la tabla 2.

Tabla 2. Valores de relación para las definiciones de sentidos de las palabras "*deposit*" y "*bank*".

Sentido elegido para *deposit*	Sentido elegido para *bank*	Valor de relación (traslape de palabras)
1	1	0
1	2	0
1	3	0
2	**1**	**2**
2	2	1
2	3	0
3	1	1
3	2	0
3	3	0

Este algoritmo tiene dos limitaciones, por un lado, la limitación principal de la medida de similitud propuesta por Lesk, es que las glosas del diccionario, regularmente, son muy cortas y no incluyen el vocabulario suficiente para identificar los sentidos relacionados [27].

Por otro lado, mientras más palabras tenga el texto, y más sentidos por cada palabra, mayor será el número de combinaciones de sentidos, haciéndolo prácticamente prohibitivo para una búsqueda exhaustiva que garantice encontrar el óptimo global exacto. Por ejemplo, para una oración de 16 palabras de contenido, donde cada palabra contiene siete sentidos (números cercanos a los observados en el corpus de SemCor), existen 7^{16} posibles combinaciones a escoger, de las cuales una será seleccionada.

Debido a estas dos limitaciones, diferentes modificaciones al algoritmo original han sido propuestas para mejorar los resultados en la desambiguación de sentidos de palabras, las cuales se describen en la siguiente sección.

4 Métodos tipo Lesk

Las modificaciones al algoritmo original de Lesk han sido propuestas tanto del lado de la medida de similitud como del lado del problema de complejidad computacional.

4.1 Basados en medidas de similitud semántica

Como se mencionó anteriormente la medida de similitud propuesta por Lesk, traslape (overlapping), tiene la limitación del tamaño de las glosas del diccionario que, regularmente son muy cortas.

A continuación se describen ciertas medidas de similitud las cuales han sido propuestas para medir la proximidad semántica entre dos sentidos o palabras, usando WordNet como espacio semántico.

Medida de Lesk Adaptada

Lesk propuso medir la similitud entre sentidos contando el traslape de palabras. La limitación principal de esta técnica es que las glosas del diccionario, por lo general, son muy breves, de tal manera que no incluyen suficiente vocabulario para identificar los sentidos relacionados. En [28] se sugiere una adaptación del algoritmo basado en WordNet. Esta adaptación consiste en tomar en cuenta las glosas de los vecinos de la palabra a desambiguar, explotando los conceptos jerárquicos de WordNet, de tal manera que las glosas de los vecinos son expandidas incluyendo a su vez las glosas de las palabras con las cuales se encuentran relacionadas mediante las diversas jerarquías que presenta WordNet. Así mismo, sugieren una variación en la manera de asignar el puntaje a una glosa, de tal manera que si "n" palabras consecutivas son iguales en ambas glosas, estas deberán de tener mayor puntaje que aquel caso en el que sólo coincide una sola palabra en ambas glosas.

Supongamos que *bark* (ladrido o corteza) es la palabra que se desea desambiguar y sus vecinos son *dog* (perro) y *tail* (cola). El algoritmo original de Lesk verifica las coincidencias en las glosas de los sentidos de *dog* con las glosas de *bark*. Luego verifica las coincidencias en las glosas de *bark* y *tail*. El sentido de *bark* con el máximo número de coincidencias es seleccionado. La adaptación del algoritmo de Lesk considera estas mismas coincidencias y añade además las glosas de los sentidos de los conceptos que se encuentran relacionados semántica o léxicamente a *dog*, *bark* y *tail*, de acuerdo a las jerarquías de WordNet.

Medida de Leacock-Chodorow

Esta medida está basada en las longitudes de rutas usando la jerarquía "es-un" de WordNet, para las definiciones de sustantivos [29]. La ruta más corta entre dos conceptos es aquella que incluye el menor número de conceptos intermedios.

Este valor es escalado por la profundidad de la jerarquía, donde dicha profundidad es definida como la longitud desde el nodo raíz hasta un nodo hoja. Por consiguiente la medida de relación está definida por la siguiente fórmula:

$$Similitud_{lch}(C_1, C_2) = \max[- \log(RutaMasCorta(C_1, C_2) /(2.D))] \quad (1)$$

$RutaMasCorta(C_1, C_2)$ es la longitud de la ruta más corta entre dos conceptos (ruta con menor número de nodos) y D es la profundidad máxima de la taxonomía (distancia entre la raíz y el nodo más alejado de ésta). La implementación de esta medida usando WordNet, asume un nodo raíz hipotético que junta todas las jerarquías de sustantivos, de tal manera que D llega a ser una constante de 16 para todos los sustantivos, lo cual significa que la longitud entre el nodo raíz y la hoja más lejana del árbol es de 16.

Medida de Resnik

Resnik [30] introduce una medida de relación basada en el concepto de "contenido de la información" más conocido en inglés como information content, el cual se trasluce

como un valor que es asignado a cada concepto en una jerarquía basada en la evidencia encontrada en un corpus.

El término "contenido de la información" es una simple medida de la especificación de un concepto. Un concepto con un gran contenido de información es muy específico a un tópico particular, mientras que conceptos con un contenido de información bajo están asociados con tópicos más generales. Por lo tanto, la expresión *carving fork* (tenedor) tiene un alto contenido de información, mientras que *entity* (entidad) tiene un bajo contenido de información.

El contenido de información de un contexto es estimado contando la frecuencia de ese concepto en un corpus de gran escala, determinando de esta manera su probabilidad. De acuerdo a Resnik, el logaritmo negativo de esta probabilidad determina el contenido de información del concepto.

$$IC(concept) = - \log(P(concept)) \quad (2)$$

Si se tuviera un texto etiquetado de sentidos, contar la frecuencia de un concepto sería logrado directamente, ya que cada concepto sería asociado con un único sentido; pero en caso contrario, Resnik sugiere contar el número de ocurrencias de una palabra en el corpus y luego dividir dicho valor por el número de sentidos que tiene dicho término, siendo este valor asignado a cada concepto. Por ejemplo, supongamos que la palabra *bank* (banco) ocurre 20 veces en un corpus, y existen dos conceptos asociados a dicha palabra en una jerarquía, uno para *river bank* (orilla de río) y el otro para *financial bank* (institución financiera). Cada uno de estos conceptos recibirá un valor de 10; en cambio si las ocurrencias de *bank*, se presentaran en un texto etiquetado con sentidos, la información sería más consistente.

La frecuencia de un concepto incluye la frecuencia de todos sus conceptos subordinados, ya que el conteo de un concepto es añadido a su inmediato superior. Es necesario notar que los conteos de los conceptos más específicos son añadidos a los más genéricos; y no de manera contraria; por ende los conteos de los conceptos específicos incrementan el total de los más genéricos. Dichos conceptos tendrán una mayor probabilidad asociada, lo que significaría que tendrían un bajo "contenido de información", ya que estos representan conceptos muy generales. La medida de Resnik usa el "contenido de información" de conceptos dentro de las jerarquías "es-un". La idea principal detrás de esta medida es que dos conceptos están semánticamente relacionados teniendo en cuenta la cantidad de información que ellos comparten en común. La cantidad de información común de dos conceptos es determinada por el "contenido de información" del concepto más bajo (*lowest common subsumer*) para las dos conceptos en cuestión. La medida de Resnik es calculada con la siguiente fórmula:

$$Similitud_{res}(C_1, C_2) = IC(lowest\_common\_subsumer(C_1, C_2)) \quad (3)$$

Esta medida no considera el contenido de información del par de conceptos a comparar y tampoco considera la longitud de la ruta entre ambos. La principal limitante de esta técnica es que algunos pares de conceptos compartirían el mismo valor de similitud, ya que existe la posibilidad de que el mismo *lowest common subsumer* sea asignado a más de un par de conceptos. Por ejemplo, *vehicle* (vehículo) es el *lowest com-*

mon subsumer de *jumbo jet* (avión jumbo), *tank* (tanque) y *house trailer* (remolque). Por consiguiente estas parejas recibirían el mismo puntaje en su comparación.

Medida de Jiang-Conrath

Jiang y Conrath [31] usan el concepto de "contenido de información" planteado por Resnik, al cual lo complementan con las longitudes de rutas entre conceptos. Esto resulta una técnica híbrida para computar la relación semántica de una pareja de conceptos. Esta técnica incluye el "contenido de información" de los propios conceptos y del *lowest common subsumer*. Esta medida es determinada por la siguiente fórmula:

$$Similitud_{jcn}(C_1, C_2) = IC(C_1) + IC(C_2) - 2 \times IC(lowest\_common\_subsumer(C_1, C_2)) \quad (4)$$

Medida de Lin

La medida de Lin [32] está basada en su teorema de similitud. Este establece que la similitud de dos conceptos es medida por la razón entre la cantidad de información necesaria para establecer la información común de ambos conceptos y la cantidad de información necesaria para describirlos. Esta información común entre dos conceptos es obtenida por el contenido de información del lowest common subsumer que aplica para ambos conceptos y el contenido de información de cada uno los conceptos propiamente dichos.

Esta medida es muy parecida a la presentada por Jiang y Conrath; aunque ellas fueros desarrolladas independientemente. Esta medida es determinada por la siguiente fórmula:

$$Similitud_{lin}(C_1, C_2) = 2 \times IC(lowest\_common\_subsumer(C_1, C_2)) / IC(C_1) + IC(C_2) \quad (5)$$

Esta medida puede ser vista como la intersección del contenido de información de los dos conceptos a comparar dividido por la suma del contenido de información de ambos.

Medida *vector*

Esta medida, al igual que la de Lesk, incorpora información de las glosas de WordNet. La medida vector, crea una matriz de co-ocurrencia para cada palabra usada en las glosas de WordNet tomando cualquier corpus y luego representa cada glosa con un vector que es el promedio de los vectores de co-ocurrencia.

Medida *path*

Este medida calcula la relación semántica de sentidos contando el número de nodos junto con el camino más corto entre el sentido en la jerarquía "es-un" de WordNet. La longitud de los caminos incluye los nodos terminales.

Dado que un camino largo indica menos relación, el valor de relación obtenido es el inverso multiplicativo de la longitud del camino (distancia) entre los dos conceptos:

relación = 1/distancia. Si los dos conceptos son idénticos, entonces la distancia entre ellos es uno; por lo tanto, su relación también es 1. Sí no es encontrado ningún camino, entonces un valor negativo muy grande es devuelto.

Medida combinada

En [33] proponen implementar una combinación de medidas de similitud dependiendo del par de categorías gramaticales a ser medidos. Ellos recomiendan utilizar la medida JCN para obtener la similitude entre sustantivos y la medida LCH para verbos. Para todas las demás relaciones se recomienda usar la medida de LESK.

4.2 Basados en la complejidad computacional

Para aliviar el problema de la complejidad computacional del algoritmo original de Lesk, dos principales soluciones han sido propuestas, a) una versión simplificada que considere las palabras, una por una, y compare cada sentido de la palabra dada con el contexto, y b) el uso de búsquedas basadas en heurísticas para encontrar una solución óptima cercana a la real en un menor tiempo [34].

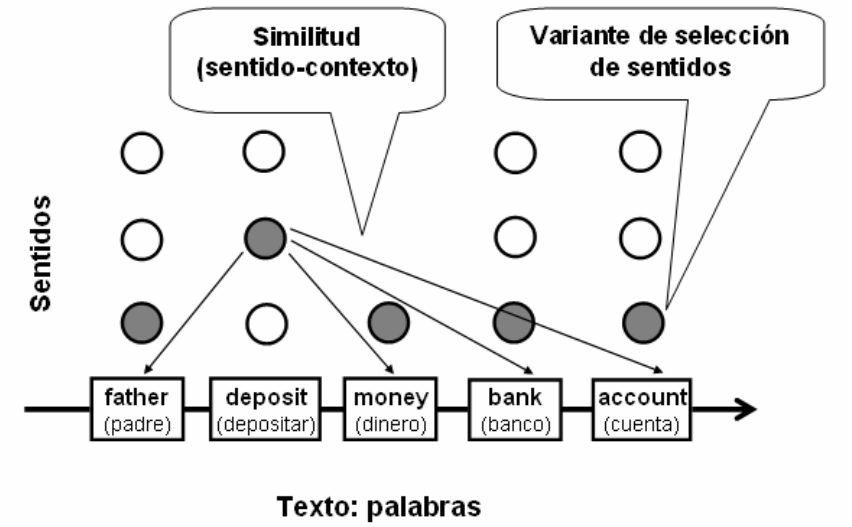

Fig. 3. Representación gráfica del algoritmo de Lesk simplificado.

Lesk simple o Lesk simplificado

Para reducir el espacio de búsqueda del algoritmo original de Lesk, Kilgarriff y Rosenzweig [35] propusieron una variación del algoritmo original de Lesk, conocido como algoritmo de **Lesk simplificado** o **Lesk simple**, donde los sentidos de las palabras en el texto son determinados uno a uno encontrando el mayor traslape entre los sentidos de las definiciones de cada palabra con el contexto actual, véase la figura 3.

En lugar de buscar asignar, simultáneamente, el significado de todas la palabras en un texto dado, este enfoque determina el sentido de las palabras uno a uno, por lo que se evita la explosión combinatoria de sentidos.

Templado simulado (Simulated Annealing)

El método de templado simulado es una técnica para la resolución de problemas de optimización combinatoria a gran escala. El nombre de este algoritmo es una analogía del proceso metalúrgico en el cuál, el metal se enfría y se templa. La característica de este fenómeno es que en el enfriamiento lento alcanza una composición uniforme y un estado de energía mínimo, sin embargo, cuando el proceso de enfriamiento es rápido, el metal alcanza un estado amorfo y con un estado alto de energía. En templado simulado la variable **T** corresponde a la temperatura que decrece lentamente hasta encontrar el estado mínimo.

El proceso requiere una función **E**, la cual representa el estado de energía de cada configuración del sistema. Es esta función la que se intenta minimizar. A grandes rasgos el algoritmo funciona de la siguiente manera: se selecciona un punto inicial y además se escoge otra configuración de manera aleatoria, se calcula para ambas configuraciones su valor **E**, si el nuevo valor es menor que el seleccionado como punto inicial, entonces el inicial es remplazado por la nueva configuración. Una característica esencial del templado simulado es que, existe el caso en el que la nueva configuración es mayor a la configuración obtenida anteriormente, y la nueva es seleccionada. Esta decisión es tomada de manera probabilística y permite salir de algún mínimo local. Una vez que el método mantenga la misma configuración por un determinado tiempo, dicha configuración es escogida como la solución.

Cowie et al. [36], basándose en el algoritmo de Lesk, utilizó este método para desambiguación de sentidos de palabras de la siguiente forma:

1. El algoritmo define una función **E** para la combinación de sentidos de palabras en un texto dado.
2. Se calcula **E** para la configuración inicial **C**, donde **C** es el sentido más frecuente para cada palabra.
3. Para cada iteración, se escoge aleatoriamente otra configuración conocida como **C'**, y se calcula su valor de **E**. Si el valor de **E** para **C'** es menor que el de **C** entonces se elige **C'** como configuración inicial.
4. La rutina termina cuando la configuración de sentidos no ha cambiado en un tiempo determinado.

Algoritmos genéticos

Introducidos por Holland [37] e impulsados en años sucesivos por Goldberg [38], uno de sus estudiantes, los algoritmos genéticos han sido utilizados con éxito en múltiples campos de la ciencia. Los algoritmos genéticos son métodos sistemáticos para la resolución de problemas de *búsqueda y optimización*, que aplican a éstos los mismos mé-

todos de la evolución biológica: selección basada en la población, reproducción sexual y mutación.

En un algoritmo genético, tras parametrizar el problema en una serie de variables, (x_i,...,x_n) se codifican en un cromosoma. Todos los operadores utilizados por un algoritmo genético se aplicarán sobre estos cromosomas, o sobre poblaciones de ellos. En el algoritmo genético va implícito el método para resolver el problema; son solo parámetros de tal método los que están codificados, a diferencia de otros algoritmos evolutivos como la programación genética. Hay que tener en cuenta que un algoritmo genético es independiente del problema, lo cual lo hace un algoritmo *robusto*, por ser útil para cualquier problema, pero a la vez *débil*, pues no está especializado en ninguno.

Las soluciones codificadas en un cromosoma *compiten* para ver cuál constituye la mejor solución (aunque no necesariamente la mejor de todas las soluciones posibles). El *ambiente*, constituido por otras soluciones, ejercerá una presión selectiva sobre la población, de forma que sólo los mejor adaptados (aquellos que resuelvan mejor el problema) sobrevivan o leguen su material genético a las siguientes generaciones, igual que en la evolución de las especies. La diversidad genética se introduce mediante mutaciones y reproducción sexual.

En la naturaleza lo único que hay que optimizar es la supervivencia, y eso significa a su vez maximizar diversos factores y minimizar otros. Un algoritmo genético, sin embargo, se usará habitualmente para optimizar sólo una función, no diversas funciones relacionadas entre sí simultáneamente. La optimización que busca diferentes objetivos simultáneamente, denominada multimodal o multiobjetivo, también se suele abordar con un algoritmo genético especializado.

Por lo tanto, un algoritmo genético consiste en lo siguiente: hallar de qué parámetros depende el problema, codificarlos en un cromosoma, y se aplican los métodos de la evolución: selección y reproducción sexual con intercambio de información y alteraciones que generan diversidad.

En [39] utilizaron un algoritmo genético que elige los sentidos que dan más coherencia al texto en términos de medidas de relación de palabras. El método optimiza globalmente el total de relaciones de palabras y no cada palabra de manera independiente [40].

Los parametros del algoritmo que utilizaron fueron los siguientes:

- El cromosoma es una secuencia de números naturales de 1 a n_i, donde n_i es el número de sentidos de la palabra w_i. Si la palabra no tiene sentidos, es decir, no se encuentra en el diccionario, la posición correspondiente en el cromosoma no se usa.
- El contenido inicial de la *piscina* fue generado aleatoriamente: para cada individuo y cada posición i en su cromosoma, el valor fue generado aleatoriamente con distribución uniforme en el dominio entre 1 y n_i.
- La función de aptitud (*fitness function*) fue definida por la siguiente fórmula:

```
for each sequence f ∈ F
  for each word w_k
      score(w_k) = Σ_i M_{i,f(w_i)}(w, f(k))
  score(f) = Σ_k^N score(w_k)
f_best = max arg (score(f))
for each word w_k
  select s_best = f_best(k)
```

- Se utilizaron dos *piscinas*, de modo que en cada generación todos los padres fueron sustituidos por sus respectivos descendientes, por lo que no hay individuos de una nueva generación que se apareen con individuos de las anteriores. Esto también significa que el método de sustitución se fijó a anexar: los nuevos individuos se anexan a la nueva *piscina*.
- No se utilizó ninguna brecha generacional, es decir, hay un número predefinido de individuos clonados en la nueva generación
- El método de selección fue *roulette wheel*: la probabilidad de seleccionar un individuo para el cruce (o clonación) es proporcional a su valor de aptitud
- El esquema de generación fue el siguiente: el par seleccionado de los padres fue reemplazado con dos descendientes formados por el intercambio de las partes seleccionadas de los cromosomas de los padres. Con cierta probabilidad de que los padres sean clonados en la nueva generación en lugar de ser acoplados.
- La probabilidad de cruce fue determinada por el parámetro llamado *crossover rate* (tasa de cruce) que controlaba la opción de cruce: si dos individuos seleccionados se cruzaban, dos descendientes se formaban como resultado o simplemente los dos padres se clonaban en la nueva generación. Entre más grande la tasa de cruce, mayor es la probabilidad de que los padres se apareen.
- El método de cruce es simple: un solo punto de cruce es seleccionado aleatoriamente; los genes incluidos hasta el punto de cruce se copian en el hijo respectivo y el resto de los genes fueron copiados a un hijo alternativo.
- El esquema de mutación es como sigue: cada hijo fue seleccionado o no para mutación con la probabilidad determinada por el parámetro llamado tasa de mutación (*mutation rate*). Si se selecciona, un único punto de mutación *i* fue seleccionado al azar(con distribución uniforme)
- Una mutación en un punto *i* fue un cambio al azar de un gen en su respectivo dominio, es decir, de 1 a n_i, donde n_i es el número de sentidos de la palabra w_i.
- Se usó elitismo para acelerar la convergencia. Esto implica las dos modificaciones siguientes en el comportamiento estándar del algoritmo. En primer lugar, dos copias del mejor individuo se clonan a la *piscina* de la nueva generación, asegurando así su supervivencia. En segundo lugar, en cada acción de cruce, de cada cuatro individuos -los dos padres y dos hijos- dos mejores se colocan en la nueva *piscina*. De esta manera, si un hijo no es tan bueno como cualquiera de los padres, no va a ser seleccionado, y uno de los padres va a sobrevivir en su lugar.
- La condición de término es la convergencia: el algoritmo se detiene cuando todos los individuos en el grupo tienen el mismo valor de aptitud (*fitness*).

5 Análisis de resultados

En esta sección se describen las evaluaciones llevadas a cabo para la desambiguación de sentidos de palabras con métodos tipo Lesk (reportados en el estado del arte) y sus principales resultados.

Para el algoritmo original de Lesk:

Lesk evaluó su algoritmo sobre ejemplos cortos extraídos de "*Pride and Prejudice"* y *"An Associated Press news story*" usando el diccionario "*Oxford Advanced Learner's Dictionary*", con una precisión de 50 a 70 %.

El trabajo de [41] presenta una evaluación del algoritmo original de Lesk con *back-off*[2] a sentido más frecuente sobre SENSEVAL-2 (*English-all words*) y SemCor (20964 instancias), utilizando *WordNet* como diccionario y con una *ventana de contexto*[3] de 2,3,8,10 y 25 palabras. Los resultados fueron similares en ambos corpus con un 43% de precisión.

Para los métodos tipo Lesk que sólo varían la medida de similitud semántica:

La medida adaptada de Lesk [28] fue evaluada sobre los datos de Senseval-2 (*English lexical sample tas*k) y utilizando el diccionario *WordNet*. En esta evaluación se utilizó una *ventana de contexto* de dos palabras y los resultados se reportan de acuerdo a la categoría gramatical de la palabra desambiguada. Para los sustantivos se reporta una precisión del 32.2%, para los verbos 24.9% y para los adjetivos 46.9%; con una precisión general de 31.7%.

En [42] evaluaron tres medidas de similitud (Jiang-Conrath, Lesk y combinada), utilizando el diccionario *WordNet*, sobre los datos de Senseval-2, Senseval-3, Semeval (*Eglish all-words data sets*) y el corpus Semcor, eliminado las oraciones con más de 210,567,168,000 combinaciones. Se utilizó una *ventana de contexto* del tamaño de la oración y evaluaron con dos *métodos de back-off*, sentido más frecuente[4] y sentido aleatorio. Los resultados experimentales mostraron que la medida combinada es más precisa que cada medida por separado. También mostraron que la medida de Lesk tiene mejor desempeño cuando el *back-off* es sentido aleatorio, mientras que la medida de Jiang-Conrath muestra mejores resultados sólo cuando se usa *back-off* a sentido más frecuente.

Para el algoritmo de Lesk simplificado:

En la propuesta original de Lesk Simple, Kilgarrif y Rosenzweig evaluaron el algoritmo sobre los datos de SENSEVAL con los sentidos obtenidos del diccionario léxi-

---

[2] Cuando método principal no tiene suficiente información para elegir el sentido de una palabra, el método de back-off toma la decisión. Ej. cuando el método de back-off es sentido aleatorio, si el algoritmo de Lesk no pudo elegir un sentido entonces se elige cualquier sentido de los posibles

[3] Número de palabras que se encuentran en el texto, antes y después de la palabra a ser desambiguada, y que se tomarán en cuenta en el proceso de desambiguación

[4] De acuerdo con WordNet.

co HECTOR. Se llevó a cabo la comparación entre sentidos utilizando por un lado la glosa del diccionario y por otro, la glosa y los ejemplos. El algoritmo presentó mejores resultados cuando se utiliza la glosa y los ejemplos, con un 55% de precisión, mientras que utilizando sólo la glosa se obtuvo un 30% de precisión.

Así mismo, en [41] se presenta también la evaluación del algoritmo de Lesk simplificado con *back-off* a sentido más frecuente sobre SENSEVAL-2 (English-all words) y SemCor (20964 instancias), utilizando WordNet como diccionario y con una ventana de contexto de 2,3,8,10 y 25 palabras. Los resultados fueron similares en ambos corpus con un 55% de precisión.

En [43] se evalúo el algoritmo de Lesk simplificado con *back-off* a sentido aleatorio sobre tres diferentes corpus usando *WordNet* como diccionario: SENSEVAL-2, SENSEVAL-3 y una muestra de 10 oraciones seleccionadas aleatoriamente del corpus Semcor. La ventana de contexto usada para los experimentos fue el tamaño de la oración y la medida de similitud usada fue el traslape pero fue normalizado, dividiendo entre largo de las definiciones, para evitar la ventaja de definiciones largas. Los resultados obtenidos fueron similares para los tres corpus con aproximadamente 48% de precisión.

Para los métodos tipo Lesk basados en heurísticas para encontrar una solución óptima en menor tiempo:

En [36] se evaluó el algoritmo de Lesk, usando templado simulado, sobre 50 oraciones etiquetadas manualmente y utilizando el diccionario LDOCE (Longman Dictionary of Contemporary English). Las oraciones tenían de dos a quince palabras, con un promedio de 5.5 palabras ambiguas por oración. En esta evaluación se reporta un 47% de precisión.

En [43] también se evalúo el algoritmo original de Lesk con templado simulado y *back-off* a sentido aleatorio sobre SENSEVAL-2, usando *WordNet* como diccionario. La ventana de contexto usada para los experimentos fue el tamaño de la oración y la medida de similitud usada fue el traslape normalizado, dividiendo entre largo de las definiciones, para evitar la ventaja de definiciones largas. Se reporta un 39.5% de precisión.

En [39] se evaluó el algoritmo de Lesk usando un algoritmo genético sobre un conjunto de 196 palabras en español obtenidas de un sitio de noticias de Internet. Como línea base de evaluación implementaron diferentes algoritmos, en todos los experimentos, el algoritmo genético mostró mejor precisión excepto por la línea base de sentido más frecuente.

6 Conclusiones

La desambiguación de sentidos de palabras es una de las tareas más importantes de la lingüística computacional o procesamiento del lenguaje natural, ya que tiene aplicación en muchas otras tareas de esta área.

Uno de los primeros algoritmos exitosos para llevar a cabo esta tarea fue el algoritmo original de Lesk, que se basa en dos ideas principales, un algoritmo de optimi-

zación y una medida de similitud para medir la relación entre las definiciones de los sentidos. El algoritmo de optimización considera la coherencia global del texto, esto es, encontrar la combinación de sentidos que maximice la relación total entre los sentidos de todas las palabras.

La principal ventaja de este algoritmo es que es un método no supervisado y sólo se necesita un diccionario de sentidos como recurso externo. Sus principales limitaciones son: con relación a la medida de similitud, que las glosas del diccionario regularmente son muy cortas y no incluyen el vocabulario suficiente para identificar los sentidos relacionados; y con relación al algoritmo de optimización, la explosión combinatoria que éste representa para un volumen de datos grande.

Por ello, diferentes métodos (conocidos como métodos tipo Lesk) han sido propuestos para aliviar estas limitaciones. En este artículo se presentó un estudio sobre estas propuestas y se presentó un análisis de los resultados que han sido obtenidos para las mismas.

Referencias

1. Bolshakov, I., Gelbukh, A.: Computational Linguistics. Models, Resources, Applications. Ciencia de la Computación. Primera Edición, México (2004)
2. Galicia-Haro, S.: Análisis sintáctico conducido por un diccionario de patrones de manejo sintáctico para lenguaje español. Tesis doctoral, CIC, IPN, México (2000)
3. Sidorov, G.: Etiquetador Morfológico y Desambiguador Manual: Dos Aplicaciones del Analizador Morfológico Automático para el Español. En: Memorias del VI encuentro internacional de computación ENC-2005, pp. 147–149, México (2005)
4. Gelbukh, A., Sidorov. G.: Procesamiento automático del español con enfoque en recursos léxicos grandes. IPN, 240 pp. (2006).
5. Weaver, W.: Translation. Mimeographed, 12 pp., July 15, 1949. Reprinted in Locke, William N. y Booth, A. Donald (1955) (Eds.), Machine translation of languages. John Wiley & Sons, pp. 15-23, New York (1949)
6. Yngve, V.: Syntax and the problem of multiple meaning. In: Locke, William N. and Booth, A. Donald (Eds.), Machine translation of languages. John Wiley & Sons, pp. 208-226, New York (1955)
7. Salton, G.: Automatic Information organization and Retrieval. McGraw-Hill, New Cork (1968)
8. Salton, G., McGill, M.: Introduction to Modern Information Retrieval. McGraw-Hill, New York (1983)
9. Krovetz, R., Croft, W.: Lexical Ambiguity and Information Retrieval. ACM Transactions on Information Systems, 10(2), pp. 115-141. (1992).
10. Voorhees, E.: Using WordNet to disambiguate word senses for text retrieval. Proceedings of the 16th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, 27 June-1 July 1993, pp. 171-180, Pittsburgh, Pennsylvania (1993)
11. Schütze, H., Pedersen, J.: Information retrieval based on word senses. Proceedings of SDAIR'95. Las Vegas, Nevada, Abril (1995).
12. Bolshakov, I., Gelbukh, A., Galicia-Haro, S.: A Simple Method to Detect and Correct Spanish Accentuation Typos. Proc. PACLING-99, Pacific Association for Computational Linguistics, Canada, pp. 104–113 (1999).

13. Yarowsky, D.: Decision lists for lexical ambiguity resolution: application to accent restoration in Spanish and French. Proceedings of the 32nd Annual Meeting of the Association for Computational Linguistics, pp. 88-95, Las Cruces, New Mexico (1994).
14. Hirst, G.: Lexical Chains as Representations of Context for the Detection and Correction of Malapropisms. WordNet An electronic Lexical Database. Edited by Christiane Fellbaum. The MIT Press. Cambridge, Massachusetts, London, England (1998)
15. Bolshakov, I., Gelbukh, A.: On Detection of Malapropisms by Multistage Collocation Testing. NLDB-2003, 8th International Conference on Application of Natural Language to Information Systems, Germany. Lecture Notes in Informatics. Bonner Köllen Verlag, pp. 28–41 (2003).
16. Bolshakov, I.A., Galicia-Haro, S.N., Gelbukh, A.: Detection and Correction of Malapropisms in Spanish by means of Internet Search. 8th International Conference Text, Speech and Dialogue (TSD-2005), Karlovy Vary, Czech Rep. Lecture Notes in Artificial Intelligence, N 3658, Springer, 2005, pp. 115–122 (2005).
17. Monroy, A., Calvo, H., Gelbukh, A.: NLP for Shallow Question Answering of Legal Documents Using Graphs. CICLing 2009. Lecture Notes in Computer Science N 5449, Springer, pp. 498–508 (2009).
18. Bhaskar, P., Pakray, P., Banerjee, S., Banerjee, S., Bandyopadhyay, S., Gelbukh, A.: Question Answering System for QA4MRE@CLEF 2012. CLEF 2012 Evaluation Labs and Workshop, Online Working Notes. Italy, 12 pp. (2012).
19. Lesk, M.: Automatic sense disambiguation using machine-readable dictionaries: how to tell a pine cone from an ice cream cone. Proc. of ACM SIGDOC Conference, p. 24–26, Toronto, Canada (1986)
20. Ledo Mezquita, Y., Gelbukh, A., Sidorov, G.: Recuperación de información con resolución de ambigüedad de sentidos de palabras para el español. Computación y Sistemas, vol. 11, no. 3, pp. 288–300 (2008).
21. Gelbukh, A., Sidorov, G., Chanona-Hernández, L.: Is word sense disambiguation useful in information retrieval? SSGRR 2003s, International Conference on Advances in Infrastructure for e-Business, e-Education, e-Science, e-Medicine, and Mobile Technologies on the Internet, track ISY, Scuola Superiore G. Reiss Romoli, L'Aquila, Italy (2003).
22. McCarthy, D., Koeling, R., Weeds, J., Carroll, J.: Finding predominant senses in untagged text. In Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics, Barcelona, Spain (2004)
23. Ríos Gaona, M.A., Godoy Calderón, S., Gelbukh, A.: Word Sense Disambiguation with the KORA-W Algorithm. Research in Computing Science, N 38, pp. 263–270 (2008).
24. Ríos Gaona, M.A., Gelbukh, A., Bandyopadhyay, S.: Web-based Variant of the Lesk Approach to Word Sense Disambiguation. Proc. of 2009 Eighth Mexican International Conference on Artificial Intelligence, IEEE CS Press, pp. 103–107 (2009).
25. Ledo Mezquita, Y., Sidorov, G., Gelbukh, A.. Tool for Computer-Aided Spanish Word Sense Disambiguation. CICLing-2003. Lecture Notes in Computer Science, N 2588, Springer, pp. 277–280 (2003).
26. Yarowsky, D.: Unsupervised word sense disambiguation rivaling supervised methods. Proceedings of ACL (1995).
27. Patwardhan, S., Banerjee, S., Pedersen, T.: Using measures of semantic relatedness for word sense disambiguation. In: Computational linguistics and intelligent text processing, pp. 241-257, Springer Berlin Heidelberg (2003).
28. Banerjee, S., Pedersen, T.: An adapted Lesk algorithm for word sense disambiguation using WordNet. In: Proceedings of the Third International Conference on Intelligent Text Processing and Computational Linguistics, Mexico City, February (2002)

29. Leacock, C., Chodorow, M., Miller, G.: Using corpus statistics and WordNet relations for sense identification. Computational Linguistics, 24(1), pp. 147-165 (1998)
30. Resnik, P.: Using information content to evaluate semantic similarity in a taxonomy. In: Proceedings of the 14th International Joint Conference on Artificial Intelligence,Montreal, August (1995)
31. Jiang, J., Conrath, D.: Semantic similarity based on corpus statistics and lexical taxonomy. In Proceedings on International Conference on Research in Computational Linguistics, Taiwan (1997)
32. Lin, D.: Using syntactic dependency as a local context to resolve word sense ambiguity. In: Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics, pages 64–71, Madrid, July (1997)
33. Sinha, R., Mihalcea, R.: Unsupervised Graph-based Word Sense Disambiguation Using Measures of Word Semantic Similarity. In: Proceedings of the IEEE International Conference on Semantic Computing (ICSC 2007), Irvine, CA, September (2007)
34. Gelbukh, A., Sidorov, G., Han, S.-Y.: On Some Optimization Heuristics for Lesk-Like WSD Algorithms. Natural Language Processing and Information Systems. 10th International Conference on Applications of Natural Languages to Information Systems, NLDB-2005, Lecture Notes in Computer Science, N 3513, Springer, pp. 402–405 (2005).
35. Kilgarriff, A., Rosenzweig, J.: Framework and results for English SENSEVAL. Computers and the Humanities, 34 (1-2), (2000)
36. Cowie, L., Guthrie, J., Guthrie, L.: Lexical disambiguation using simulated annealing. COLING (1992)
37. Holland, J. H.: Adaptation in Natural and Artificial Systems. University of Michigan Press, Ann Arbor (1975).
38. Goldberg D.E.: Genetic algorithms in search, optimization, and machina learning, Addison-Wesley, New York, (1989)
39. Gelbukh, A., Sidorov, G., Han, S.: Evolutionary Approach to Natural Language Word Sense Disambiguation through Global Coherence Optimization. WSEAS Transactions on Communications, Issue 1 Vol. 2, p. 11–19 (2003)
40. Gelbukh, A., Han, S.-Y., Sidorov, G.: Comparison of some global coherence optimization heuristics for word sense disambiguation. Avances en: Ciencias de la Computación, CIC-2003, XII Congreso Internacional de Computación, pp. 131–135 (2003).
41. Vasilescu, F., Langlais, P., Lapalme, G.: Evaluating variants of the Lesk approach for disambiguating words, LREC (2004)
42. Torres, S., Gelbukh, A.: Comparing similarity measures for original WSD lesk algorithm. Advances in Computer Science and Applications Research in Computing Science, 43, pp-155-166, México (2009)
43. Mihalcea, R.: Unsupervised large-vocabulary word sense disambiguation with graph-based algorithms for sequence data labeling. In Proceedings of HLT05, Morristown, NJ, USA (2005)