Desambiguación de sentidos de palabras usando relaciones sintácticas como contexto local

Javier Tejada-Cárcamo,[1,2] Alexander Gelbukh,[1] Hiram Calvo[1]

[1] Natural Language Processing Laboratory,
Center for Computing Research, National Polytechnic Institute,
Mexico City, 07738, Mexico

[2] Peruvian Society for Computer Science, Peru

jawitejada@hotmail.com, gelbukh@gelbukh.com,
likufanele@likufanele.com; www.Gelbukh.com

Abstract. La desambiguación automática de sentidos de palabras es una tarea necesaria para muchas otras operaciones en el procesamiento de lenguaje natural. Inspirados por el trabajo de McCarthy *at al*. (2004), presentamos una variante de su método que desambigua una ocurrencia específica de una palabra dada y no tan sólo encuentra el sentido más frecuente de esta palabra. Aunque el trabajo está en progreso, los resultados son prometedores.

1. Introducción

La desambiguación automática de sentidos de palabras se ha convertido actualmente en una tarea intermedia (tarea complementaria para muchas otras operaciones [5, 7]) y necesaria para muchas de las áreas que incurren en el tratamiento del lenguaje natural, tales como traducción automática, recuperación de información, análisis temático y de contenido, análisis gramatical, procesamiento de habla, procesamiento de texto, etc. Es por esto, que la importancia de saber el sentido de una palabra es fundamental para poder completar, deducir o inducir resultados en las áreas mencionadas anteriormente.

El objetivo principal de este trabajo, es exponer un nuevo método para la desambiguación automática de sentidos de palabras.

La premisa en la que se basa el método propuesto es: dos palabras diferentes tienen significados similares si ocurren en contextos locales idénticos. Consideremos la siguiente oración:

The new facility will employ 500 of the existing 600 employees

La palabra *facility* tiene 5 posibles significados en WordNet 2.0 [2]: *installation*, *proficiency* / *technique*, *adeptness*, *readiness*, *toilet*/*bathroom*. Encontramos palabras que aparecieron en el mismo contexto que *facility*, usando una base de datos de dependencias sintácticas recopiladas de WordNet 2.0, las cuales se muestran en la tabla 1.

(C) Alexander Gelbukh, Manuel Montes y Gómez (Eds.)
Advances in Natural Language Understanding and Intelligent Access to Textual Information,
Proceedings of NLUIATI-2005 workshop in conjuction with MICAI-2005, p. 31-40.

Tabla 1: Palabras similares a *facility*

Palabra	Frecuencia
Plant	9
Company	8
Operation	7
Industry	7
Firm	6
Unit	6
Enterpise	6
Corporation	5
Machina	4
Department	3
Pilot	2
Manufacturer	1

El sistema propuesto para desambiguar sentidos de palabras en forma automática está dividido en tres partes descritos a continuación.

2. Creación de la base de datos de recursos sintácticos

La base de datos de recursos sintácticos está conformada por diferentes tuplas sintácticas, cada una de las cuales comprenden una palabra principal, la que llamaremos cabeza, y un conjunto de palabras secundarias, las que llamaremos colas, que dependen sintácticamente (gramáticas de dependencia) de la cabeza, en el marco de una oración específica. De esta manera, la tupla de una cabeza, podría ser representada por una lista de colas:

$$Tupla\ (cabeza) = \{cola\_1, cola\_2, cola\_3, ..., cola\_n\}$$

Las gramáticas de dependencia representan las oraciones como un conjunto de relaciones de dependencia sintáctica, las cuales forman un árbol que conectan todas las palabras en una oración. Una palabra en una oración puede tener diferentes modificadores (colas), sin embargo sólo puede modificar a una palabra (cabeza). La raíz del árbol de dependencia no modifica a ninguna palabra y puede ser llamada la cabeza de la oración [3]. La obtención de estas relaciones para la creación de nuestras tuplas, fueron obtenidas usando el analizador sintáctico MINIPAR [4], el cual logra resultados de hasta 80% de efectividad en la obtención de este tipo de relaciones. Así mismo, el corpus que se usó, fue el de SEMCOR 2.0, del cual extrajimos sus oraciones y las parseamos con MINIPAR, obteniendo árboles de constituyentes, de ellos los árboles de dependencia y, finalmente, las mencionadas relaciones de dependencia sintáctica. Por ejemplo:

He likes the woman very much

Las relaciones de dependencia obtenidas serían:

$$Tupla\ (Like) = \{he, woman, very\ much\}$$
$$Tupla\ (woman) = \{the\}$$

De esta manera, a medida que se van parseando las diferentes oraciones, los elementos de cada tupla se van incrementando o repitiendo.

Es necesario aclarar que en la construcción del recurso sintáctico se tuvo un tratamiento especial que puede ser resumido en 3 casos:

a. *Con las preposiciones*, ya que de acuerdo a nuestro criterio, éstas no aportan mucha semántica por sí mismas. A continuación se muestra un fragmento de una oración:

   The jury further said in term end presentments that the City_Executive_Committee, which had over-all ***charge of the election****…*

   El analizador sintáctico (en adelante parser), al parsear la parte resaltada en negrita nos muestra el siguiente resultado:

```
18     (charge      ~ V    E1     i      (gov fin))
E9     (()    overall N    18     subj   (gov charge)
       (antecedent 17))
19     (of    ~ Prep 18     mod    (gov charge))
20     (the   ~ Det  21     det    (gov election))
21     (election    ~ N    19     pcomp-n (gov of))
```

   Si obtuviéramos las tuplas de dependencia sintáctica de la manera tradicional tendríamos tres tuplas, tal como se muestra a continuación:

   Tupla (Charge) = {*of*}
   Tupla (of) = {*election*}
   Tupla (election) = {*the}*

   En cambio, si hubiéramos obtenido las tuplas de dependencia sintáctica sin tener en cuenta las preposiciones:

   Tupla (Charge) = {*election*}
   Tupla (election) = {*the*}

   Al comparar las diferentes tripletas generadas por ambos métodos, se puede apreciar que el caso en el cual se "ignoran las preposiciones" tiene más semántica que el anterior. Esta modificación es más notoria en los verbos, ya que muy frecuentemente estos siempre son modificados por frase preposicionales.

b. Asimismo, otras de las características que es necesario resaltar en la construcción de nuestro recurso sintáctico es que incluimos como parte del conjunto de modificadores de la cabeza, *la palabra a la cual modifica la cabeza*, lo cual no es muy usual en la construcción de este tipo de recursos, pero la idea principal en la construcción de este recurso es conseguir tuplas que tenga fuerte relación sintáctica y semántica. Observemos el siguiente fragmento de oración:

   It urged that the city take steps to ***remedy this problem****.*

   El parser nos dará el siguiente resultado en lo que respecta al texto en negritas:

```
11     (remedy      ~ V    E0     i      (gov inf))
E10    (()    it N   11     subj   (gov remedy)  (antecedent E5))
14     (this  ~ Det  15     det    (gov problem))
15     (problem ~ N 11     obj    (gov remedy))
```

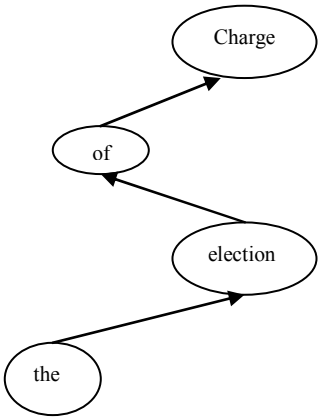

Grafico 1: Dependencia Sintáctica con preposición

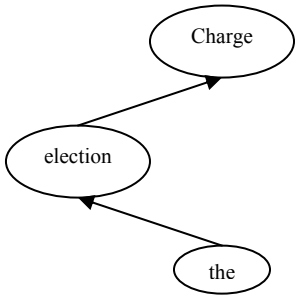

Grafico 2: Dependencia Sintáctica sin preposición

Si obtuviéramos las tuplas de dependencia sintáctica de la manera tradicional tendríamos dos tuplas, tal como se muestra a continuación :

Tupla (*remedy*) = {*problem*}
Tupla (*problem*) = {*this*}

Si obtuviéramos las tuplas de dependencia sintáctica, incluyendo como parte de los modificadores la palabra a la cual modifica la cabeza obtendríamos:

Tupla (*remedy*) = {*problem*}
Tupla (*problem*) = {*this*, *remedy*}

Como se puede observar el conjunto de elementos *Tupla (problem)* tiene más semántica en el segundo caso que en el primero, ya que el hecho de ver en el primer caso que la palabra *remedy* es modificado por *this*, no aporta nada excluyente en nuestro recurso, es decir, *this* podría modificar a cualquier sustantivo, en cambio en el segundo caso vemos como parte de los modificadores de *problem* la palabra *remedy,* lo cual hace que la tupla tenga mayor peso semántico.

c. *Cabezas iguales léxicamente.* Es necesario notar la posibilidad de encontrar cabezas iguales léxicamente, que tengan diferentes POS (*part of speech*) o diferentes sentidos, lo cual hace que sean cabezas diferentes, tal como se muestra a continuación:

I ***dictate*** *a letter*
Tupla (*Dictate*) = {*I, letter*}

I followed the ***dictates*** *of my conscience*
Tupla (*Dictate*) = {*The, of*}

El *wordterm Dictate* puede ser verbo o sustantivo, de tal manera que ambas cabezas serán tomadas como diferentes, ya que una misma palabra puede desempeñar

diferentes roles sintácticos en una oración, y suponemos que los contextos de un mismo *wordterm* que tiene diferentes POS suelen ser diferentes.

La información necesaria para obtener los POS, son proporcionadas por los resultados que ya vienen con SEMCOR y además, por los resultados obtenidos por MINIPAR al haber procesado las oraciones de dicho corpus. De esta manera, se tiene dos fuentes de información al respecto, las que permitirán discriminar al menos sintácticamente aquellas cabezas cuya POS difiera en ambos recursos.

3. Módulo de Consulta

A continuación se describe el módulo de consulta de palabras similares usando nuestro Sistema de Recuperación de Información (SRI).

El objetivo principal de nuestro SRI, es obtener palabras similares (figura 3), las cuales suelen usarse en contextos similares al de la palabra que se quiere desambiguar. A estas palabras la llamaremos de ahora en adelante *vecinos*.

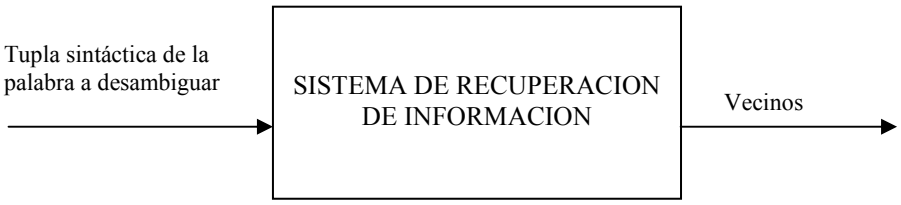

Gráfico 3: Entradas y salidas del SRI

Nuestro SRI está basado en el Modelo de Espacio Vectorial –el esquema TF-IDF [1]–, el cual generalmente es aplicado para tareas de clasificación y similitud de documentos, representando cada documento por un vector y comparándolos usando esquemas multidimensionales; sin embargo para nuestra tarea, un documento equivaldrá a una tupla, como las que mencionamos anteriormente.

Al finalizar el proceso anterior, obtuvimos cerca de 50.000 tuplas en nuestra base de datos de recursos sintácticos, las cuales tienen el siguiente formato:

Tupla (*cabeza*) = {(*cola_1*,*frec*); (*cola*_2, *frec*); (*cola*_3, *frec*); ...; (*cola_n, frec*)}

Como parte del modelo del Espacio Vectorial, es necesario calcular la Frecuencia Normalizada del Término (TF) que en nuestro caso sería de una cola, la Frecuencia Inversa del Documento (IDF) que en nuestro caso se refiere a la *tupla,* y el Peso del Termino (*w*).

La Frecuencia Normalizada(TF) de una cola se calcula con la siguiente fórmula:

$$f_{i,j} = \frac{freq_{i,j}}{\max freq_{l,j}}$$

donde:

$freq_{i,j}$ = Frecuencia de la cola i en la cabeza j.

$\max freq_{l,j}$ = Máxima frecuencia de las colas que pertenecen a esa cabeza.

La Frecuencia Inversa de una cola se calcula con la siguiente fórmula:

$$idf_i = \log \frac{N}{n_i}$$

donde:

N = Número total de cabezas en el sistema.
n_i = Número de cabezas en la cuales aparece la cola i.

Finalmente, el Peso de una cola lo calculamos con la siguiente fórmula :

$$w_{i,j} = f_{i,j} \times \log \frac{N}{n_i}$$

El TF, muestra la importancia de una cola respecto a la cabeza que modifica, de tal manera, que el peso de una cola respecto a una cabeza, ***aumenta*** si ésta aparece más a menudo con dicha cabeza, sucediendo lo contrario con el IDF, que muestra la importancia de una cola respecto al resto de cabezas de nuestro recurso, de tal manera que el peso de una cola ***disminuye*** si aparece más a menudo en todas las demás cabezas, ya que las colas muy frecuentes en otras cabezas discriminan poco a la hora de representar la cabeza mediante un vector.

Al final de calcular las fórmulas mencionadas, cada cabeza es representada en nuestro SRI por un conjunto de tuplas cuaternarias:

$$Tupla(i) = \{(cola_1, tf_1, idf_1, w_1), (cola_2, tf_2, idf_2, w_2) \ldots . (cola_n, tf_n, idf_n, w_n)\}$$

Tal como se ve en la Figura 3, nuestro SRI recibe como entrada una tupla, que se encuentra en función de la palabra que se desea desambiguar, la cual se representará por un vector (q), y cada tupla de nuestro recurso se representará por un vector (vj), de tal manera que es muy fácil intuir que el número de dimensiones del vector vj, es mucho mayor al número de dimensiones del vector q, ya que en varios casos las dimensiones de q oscilan entre 1 y 5 dimensiones como máximo, mientras que las dimensiones de los vectores del recurso varían entre 1 y 13.500. El SRI se encarga de comparar el vector entrante con los aproximadamente 50.000 vectores que dispongamos en nuestro recurso, siendo necesario para ello igualar previamente las dimensiones de ambos vectores rellenando de 0 (ceros) aquellas dimensiones que no tengan valor para cualquiera de los dos vectores a comparar; de esta manera construímos un sistema de vectores de 'n' dimensiones.

El modelo de vectores propuesto para evaluar el grado de similitud de 'q' y 'vj' puede ser cuantificado por el coseno del ángulo entre estos dos vectores, como lo muestra la siguiente expresión:

$$\text{Valor} = \frac{\vec{v}_j \bullet \vec{q}}{\left|\vec{v}_j\right| \times \left|\vec{q}\right|} = \frac{\sum_{i=1}^{t} w_{i,j} \times w_{i,q}}{\sqrt{\sum_{i=1}^{t} w_{i,j}^2} \times \sqrt{\sum_{i=1}^{t} w_{i,q}^2}}$$

De esta manera, al comparar '*q*' con cada una de los vectores de nuestro recurso se obtendrán los vecinos de '*q*', es decir, aquellas palabras que son usadas en el mismo contexto sintáctico (palabras similares de acuerdo a nuestra hipótesis) que la palabra que se quiere desambiguar. El valor de comparación entre los vectores oscila entre 0 y 1, de tal manera que el SRI tiene como parámetro configurable el umbral sobre el que se tomaran los vecinos que satisfagan dicho límite. A continuación, se muestra un ejemplo de los procesos que se han explicado hasta el momento. Para ello damos la siguiente oración:

> *Pelham said Sunday night there was research being done on whether the quickie vote on the increase can be repealed outright or whether notice would have to first be* ***given*** *that reconsideration of the action would be sought*

La palabra que se desea desambiguar es aquella que esta resaltada en negrita, de tal manera que el parseador nos da la siguiente respuesta:

```
> (
E6  (()    fin C  *      )
1   (Pelham      ~ N    2      s      (gov say))
2   (said  say V  E6     i      (gov fin))
E10 (()    Pelham N      2      subj (gov say) (antecedent 1))
E2  (()    fin C  2      fc     (gov say))
3   (Sunday      _DATE N       5      nn     (gov there))
4   (night _TIME N       5      nn     (gov there))
5   (there ~ N    6      s      (gov be))
6   (was   be VBE E2     i      (gov fin))
7   (research    ~ N     6      pred   (gov be))
E11 (()    there N       7      subj (gov research) (antecedent 5))
E1  (()    vpsc C 6      mod    (gov be))
E8  (()    ~ N    E1     s      (gov vpsc)    (antecedent 5))
8   (being be be  9      be     (gov do))
9   (done  do V   E1     i      (gov vpsc))
E12 (()    ~ N    9      obj    (gov do)      (antecedent E8))
10  (on    ~ Prep 9      mod    (gov do))
E0  (()    fin C  10     pcomp-c       (gov on))
11  (whether     ~ COMP E0     c      (gov fin))
12  (the   ~ Det  18     det    (gov vote))
15  (quickie     ~ N    18     nn     (gov vote))
18  (vote  ~ N    24     s      (gov repeal))
19  (on    ~ Prep 18     mod    (gov vote))
20  (the   ~ Det  21     det    (gov increase))
21  (increase    ~ N    19     pcomp-n       (gov on))
22  (can   ~ Aux  24     aux    (gov repeal))
23  (be    ~ be   24     be     (gov repeal))
24  (repealed    repeal V      E0     i      (gov fin))
E13 (()    vote N 24     obj    (gov repeal)  (antecedent 18))
25  (outright    ~ A    24     mod    (gov repeal))
E5  (()    fin C  E6     conj   (gov fin))
E4  (()    fin C  E5     mod    (gov fin))
27  (whether     ~ COMP E4     c      (gov fin))
28  (notice      ~ N    30     s      (gov have))
```

```
29 (would ~ Aux  30     aux    (gov have))
30 (have  ~ V    E4     i      (gov fin))
E14 (()   notice N     30     subj (gov have) (antecedent 28))
E3 (()    inf C  30     fc     (gov have))
E7 (()    ~ N    E3     s      (gov inf)    (antecedent 28))
31 (to    ~ Aux  34     aux    (gov give))
32 (first ~ A    34     guest  (gov give))
33 (be    ~ be   34     be     (gov give))
34 (given give V E3     i      (gov inf))
E15 (()   ~ N    34     obj    (gov give)   (antecedent E7))
35 (that  ~ N    34     obj2   (gov give))
36 (reconsideration    ~ N    42     s      (gov seek))
37 (of    ~ Prep 36     mod    (gov reconsideration))
38 (the   ~ Det  39     det    (gov action))
39 (action       ~ N    37     pcomp-n      (gov of))
40 (would ~ Aux  42     aux    (gov seek))
41 (be    ~ be   42     be     (gov seek))
42 (sought       seek V E5     i      (gov fin))
E16 (()   reconsideration N   42     obj    (gov seek)
          (antecedent 36))
)
```

La tupla de entrada para el SRI sería:

$$Tupla\ (give) = \{to, first, be, that\}$$

Y después de representar esta tupla mediante un vector y compararla con los 50.000 vectores de nuestro recurso mostramos sólo los 10 primeros vecinos que nos devuelve el SRI.

```
64;5;give#v;compliment#v; 0.77719297123081799938124205 59;
      elucidate#v;        0.77719297123081799938124205 59;
      furlough#v;         0.77719297123081799938124205 59;
      purge#v;            0.77719297123081799938124205 59;
      subjected#v;        0.62715376443546733185971318 84;
      appealed#v;         0.61796263577535704434608366 06;
      institutionalize#v; 0.61796263577535704434608366 06;
      affix#v;            0.61082120009302840822891070 35;
      Plank#v;            0.57735026918962635074788945 19;
      blister#v;          0.57735026918962486235163089 43;
```

4. Módulo de Comparación de Sentidos

Este módulo se basa principalmente en el método propuesto por Diana McCarthy *et al*. [6], con dos diferencias:

a. El método de McCarthy es usado para encontrar el sentido más predominante de una palabra, para lo cual toma como entrada los vecinos que obtiene del Tesauro de Lin [3] para una palabra determinada.

Nuestra propuesta usa el método de McCarthy para obtener no el sentido mas frecuente de una palabra; sino para determinar el sentido que le corresponde a dicha palabra en un contexto específico.

b. El método propuesto no usa la ponderación que hace Lin para cada vecino, sino que usamos el valor que nos devuelve nuestro SRI, el cual a diferencia del de Lin, pondera los vecinos de acuerdo a un contexto local específico.

Es necesario obtener un peso para cada sentido de la palabra a desambiguar teniendo en cuenta los vecinos obtenidos en el proceso anterior. Para ello la fórmula que usamos para obtener el peso de cada sentido es la siguiente:

$$\sum_{n_j \in N_w} SRI(w, n_j) x \frac{wnss(ws_i, n_j)}{\sum_{ws_{i'} \in senses(w)} wnss(ws_{i'}, n_j)}$$

$$wnss(ws_i, n_j) = \max_{ns_x \in senses(n_j)} (wnss(ws_i, ns_x))$$

La función *wnss* que realiza la comparación entre dos sentidos, usa el paquete *WordNet::Similarity* de Siddharth Patwardhan, Ted Pedersen, and Jason Michelizzi, el cual contiene diversas medidas de similitud, de las cuales sólo hemos usado las de JCN, LESK y LIN.

La Tabla 2 muestra los resultados obtenidos al procesar 150 palabras tomadas en forma aleatoria de WordNet.

Tabla 2: Resultados finales usando 3 diferentes medidas

Medida	Precision (%)
JCN	63 %
LESK	65 %
LIN	60 %

5. Conclusiones

Hemos presentado un método para la desambiguación de sentidos de palabras tomando en cuenta la similitud entre palabras que se usan en contextos similares. Nuestro trabajo fue inspirado por las ideas de McCarthy *et al*. [6]. Sin embargo, los últimos autores tan sólo aplicaron su idea a la determinación del sentido más frecuente en general en el lenguaje. A diferencia, nuestra intención era aplicar la misma idea para desambiguar el sentido específico en un contexto dado, sin promediar por todas las ocurrencias de la palabra dada en todos los textos de un corpus.

Aunque nuestro trabajo está en progreso, los resultados presentados en la tabla 2 se ven muy prometedores. Esperamos mejorarlos con una selección más fina de los rasgos de las relaciones sintácticas útiles para nuestro método: por ejemplo, ignorar las palabras sin sentido, ignorar ciertos tipos de relaciones sintácticas, etc. Otra idea sería probar otros tipos de contextos: coocurrencias dentro de la misma oración o del mis-

mo párrafo, distancia lineal entre las palabras, distancia entre las palabras en el árbol sintáctico, etc., y combinar los resultados obtenidos con diferentes tipos de contexto.

Referencias

1. R. Baeza-Yates and B. Ribeiro-Neto. Modern Information Retrieval. Addison-Wesley, 1999.
2. C. Fellbaum (ed.). WordNet, An Electronic Lexical Database. MIT Press, 1998, 423 pp.
3. D. Lin. Automatic retrieval and clustering of similar words. *COLING-ACL 98*, Montreal, Canadá, 1998.
4. D. Lin. Dependency-based Evaluation of MINIPAR. Workshop on the Evaluation of Parsing Systems, Granada, España, 1998.
5. C. Manning and H. Schütze, *Foundations of Statistical Natural Language Processing*. MIT Press, 1999.
6. D. McCarthy, R. Koeling, J. Weeds, and J. Carroll. McCarthy Finding Predominant Word Senses in Untagged Text. *ACL-2004*, Barcelona, España, 2004, p. 279–286.
7. Y. Wilks and M. Stevenson. The grammar of sense: Is word-sense tagging much more than part-of-speech tagging? *Technical Report CS-96-05*, University of Sheffield, 1996.