Actas del

VIII Congreso Español sobre Tecnologías y Lógica Fuzzy

Pamplona, 8-10 de septiembre de 1998

Universidad Pública
de Navarra
Nafarroako
Unibertsitate Publikoa

EUSFLAT

EUROPEAN SOCIETY
FOR FUZZY LOGIC
AND TECHNOLOGY

Título: *Actas del VIII Congreso Español sobre Tecnologías y Lógica Fuzzy*

Editor Científico: Pedro Burillo López

Comité Editorial: Gregorio Buldain
Ana Burusco
Humberto Bustince
Noé Frago
Remón Fuentes
Ángel Marín
Mª Victoria Mohedano

Editan: Universidad Pública de Navarra. Departamento de Automática y Computación
European Society for Fuzzy Logic and Technology

ISBN: 84-95075-10-5

D.L.: NA 1.698/1998

Fotocomposición: Página S. L.

Imprime: Digitalia. Impresión digital

Impreso en papel ecológico

La Caja de Ahorros de Navarra ha financiado la publicación de esta obra.

Estas actas han sido impresas mediante un proceso digital, que reproduce íntegramente los originales suministrados por los autores, incluidas sus características tipográficas. Por este motivo en el presente volumen aparecen distintas familias de letras.

Coordinación y distribución: Dirección de Publicaciones
Universidad Pública de Navarra
Campus de Arrosadía
31006 Pamplona
Fax: (948) 16 93 00
E-mail: publicaciones@upna.es

Índice

ANÁLISIS DEL GRADO DE SINONIMIA EN UN DICCIONARIO DE SINÓNIMOS

Santiago Fernández Lanza
Área de Lógica y Filosofía de la Ciencia
Campus Sur s/n
15706-Santiago de Compostela
e-mail: sflanza@usc.es

Alejandro Sobrino Cerdeiriña
Área de Lógica y Filosofía de la Ciencia
Campus Sur s/n
15706-Santiago de Compostela
e-mail: lflgalex@usc.es

Resumen

En este trabajo se propone un programa Prolog que analiza el grado de sinonimia que poseen dos palabras en un diccionario de sinónimos y su comportamiento cuando son sustituidas en una oración.

Palabras Clave: Sinonimia, diccionario de sinónimos, procesamiento del lenguaje natural, Prolog.

1 INTRODUCCIÓN

Si pretendemos caracterizar la sinonimia tal y como la concibe un humano, inevitablemente deberemos reparar en aspectos como la contextualidad, la subjetividad, la ambigüedad, la imprecisión o el carácter gradual presente en la misma. Ante tal variedad y complejidad de aspectos involucrados, este trabajo pretende hacer, en vez de un análisis teórico de la sinonimia, un estudio empírico. Dicho estudio se basará en el análisis de un diccionario de sinónimos. Se utilizará el *Diccionario de sinónimos y antónimos de la lengua española* de Samuel Gili Gaya [4]. En el cual, la información relativa a la sinonimia, se estructura de la siguiente forma: palabra, lista de palabras sinónimas, acepción. Por ejemplo, la palabra "concesión" posee la lista de sinónimos {"permiso", "licencia", "gracia", "privilegio"} en la primera acepción y {"epítrope"} en la segunda.

El objetivo usual del manejo de un diccionario de este tipo es buscar los sinónimos de una palabra para sustituirla por otra en una oración. Ahora bien, si concebimos la sinonimia como una cuestión de grado, la sustitución conlleva el riesgo de que la nueva palabra no signifique exactamente lo mismo, y esto puede afectar al contenido informativo de la oración completa.

Una característica importante de los diccionarios de sinónimos es que para definir un término se utiliza una simple lista de palabras. Por tanto, para ver si dos términos están relacionados, se comparan dos listas de palabras. Si la sinonimia es una cuestión de grado y se representa como una lista el conjunto de sinónimos de una palabra, la cuestión que se plantea a continuación es cómo medir el grado de sinonimia entre dos palabras. En [5] se establece un coeficiente que se interpreta como la asociación entre dos listas de sinónimos. Utilizaremos este coeficiente para medir el grado de sinonimia GRS(a, b) entre dos palabras a y b:

$$GRS(a,\ b) = \frac{|A \cap B|}{|A \cup B|}$$

donde A y B son, respectivamente, las listas de sinónimos de a y b. El tratamiento en Prolog del lenguaje natural por medio de listas, permitirá hallar mecánicamente este grado.

En el contexto de un diccionario de sinónimos caben plantearse distintos aspectos relacionados con el tema que nos ocupa y que serán tenidos en cuenta en la elaboración del trabajo:

1.- El problema de la polisemia. Las palabras pueden poseer varios significados o acepciones según el contexto de uso. Esto hace que podamos comparar cada palabra teniendo en cuenta el conjunto genérico de sus sinónimos o bien solamente acepciones concretas, adecuadas a un uso determinado. Esta propuesta parece más oportuna, ya que evita la interferencia de otras acepciones en la decisión de la sinonimia de dos palabras cuando se utilicen en un contexto concreto.

2.- La reflexividad de la sinonimia. Los diccionarios de sinónimos no incluyen, evidentemente, el caso trivial de que, fijada una acepción, una palabra sea sinónima de sí misma. Sin embargo, este hecho puede llevar a situaciones como ésta: en [4] figura solamente la palabra "pernil" como único sinónimo de "pernera" y la palabra "pernera" como único sinónimo de "pernil" en la segunda acepción. Si comparamos ambas palabras según esas acepciones, la intersección de ambos conjuntos de sinónimos es vacía, con lo que el grado de sinonimia medido con la fórmula expresada anteriormente sería 0, resultado contraintuitivo. Una forma de evitar esto es incluir el caso trivial de la palabra idéntica en el conjunto de sinónimos de una palabra. En nuestro ejemplo, el conjunto de sinónimos de "pernera" sería {"pernera", "pernil"} y el de la segunda acepción de "pernil" {"pernil", "pernera"}; de esta forma el resultado sería, como cabría esperar, que son sinónimas en grado 1.

3.- La simetría de la sinonimia. Es usual atribuir la propiedad simétrica a la sinonimia, pero si observamos un diccionario de sinónimos, en muchos casos dicha propiedad no se da; es decir, a veces una palabra A tiene a otra B en la lista de sus sinónimos, pero no a la inversa.

2 SINONIMIA ENTRE PALABRAS

A continuación presentaremos un programa Prolog que calcula el grado de sinonimia entre las palabras de un diccionario de sinónimos. Los ejemplos pertenecen a [4]. La medida se realiza acepción por acepción, de tal forma que dadas dos palabras A y B, el compilador fijará la primera acepción de A y comprobará que B pertenece a la lista de sinónimos de A en esa acepción, calculará el grado de sinonimia que posee A en esa acepción con todas las acepciones de B y seleccionará la acepción de B cuyo grado de sinonimia sea mayor. Después repetirá el mismo proceso con las demás acepciones de A hasta agotarlas.

Se puede ver un diccionario de sinónimos como una base de datos de palabras (entradas) y listas de palabras (sinónimos) agrupadas por acepciones. Utilizaremos el siguiente predicado:

`sin_dic(Palabra, Lista_sinónimos, Acepción)`

para representar la información del diccionario. El primer argumento del predicado `sin_dic` es la entrada del diccionario, el segundo la lista de sus sinónimos y el tercero la acepción que corresponde al grupo de sinónimos. Veamos un ejemplo, que recoge varias entradas de [4]:

```
%SINÓNIMOS DEL DICCIONARIO

sin_dic(concesion,[concesion,permiso,licencia,gr
acia,privilegio],1).
sin_dic(concesion,[concesion,epitrope],2).

sin_dic(permiso,[permiso,autorizacion,consentimi
ento,licencia,venia,beneplacito,aquiescencia],1)
.

sin_dic(licencia,[licencia,permiso,autorizacion,
consentimiento,venia,concesion],1).
sin_dic(licencia,[licencia,abuso,osadia,atrevimi
ento,desenfreno,libertinaje],2).

sin_dic(gracia,[gracia,beneficio,favor,merced,do
n,regalo],1).
sin_dic(gracia,[gracia,perdon,indulto],2).
sin_dic(gracia,[gracia,benevolencia,amistad,afab
ilidad,agrado],3).
sin_dic(gracia,[gracia,garbo,donaire,sal,salero,
angel,atractivo,encanto],4).
sin_dic(gracia,[gracia,chiste,agudeza,ocurrencia
],5).

sin_dic(privilegio,[privilegio,prerrogativa,conc
esion],1).

sin_dic(epitrope,[epitrope,permision],1).
sin_dic(epitrope,[epitrope,concesion],2).
```

El procedimiento general para calcular el grado de sinonimia entre dos palabras consiste en los siguientes pasos. Dadas dos palabras A y B:

1.- Tómese la lista de sinónimos de A en la primera acepción, a lo que llamaremos ListaA.

2.- Compruébese que B pertenece a ListaA.

2.1.- Si no pertenece a ListaA, realizar el paso 1 con la siguiente acepción de A. Si ya se comprobaron todas las acepciones de A y la negativa persiste, entonces A y B no son sinónimas.

2.2.- Si pertenece a ListaA, pasar a 3.

3.- Tómese la lista de sinónimos de B en la primera acepción ListaB y calcúlese el grado mediante la fórmula:

$$\text{GradoAcep1} = \frac{|\text{ListaA} \cap \text{ListaB}|}{|\text{ListaA} \cup \text{ListaB}|}$$

4.- Repítase el paso 3 con todas las acepciones de B.

5.- Calcúlese el máximo de los grados obtenidos en 4 y su correspondiente acepción. Este grado máximo es el grado de sinonimia entre A y B.

El predicado `sinónimo` calcula el grado de sinonimia entre dos palabras:

`sinónimo(A, AcepA, B, AcepB, GRS)`

este predicado se puede leer como, la palabra A en la acepción AcepA es sinónima de la palabra B en la acepción AcepB con grado GRS.

El predicado se define de la siguiente forma:

1.- Cualquier palabra es sinónima de sí misma en grado 1 si pertenece al diccionario y su acepción no varía.

2.- Dos palabras distintas son sinónimas en un grado GRS si

- fijada una acepción de la primera, la segunda pertenece al listado de sinónimos de aquella, y
- Lista1 es la lista de todos los grados que se pueden obtener variando todas las acepciones de la segunda palabra, y
- Lista2 es la lista de todas las acepciones que corresponden respectivamente a los grados calculados en el punto anterior, y
- GRS_MAX es el máximo de los grados de Lista1 y AcepB es la acepción correspondiente a ese grado.

El código es el siguiente:

```
%SINONIMIA ENTRE PALABRAS
sinonimo(A,X,A,Y,1.0):-
  sin_dic(A,Z,X),
  X=Y.
sinonimo(A,AcepA,B,AcepB,GRS_MAX):-
  comparable(A,B,AcepA),
  findall(GRS,sin_acep(A,AcepA,B,Acep,GRS),Lista1),
  findall(Acep,sin_acep(A,AcepA,B,Acep,GRS),Lista2),
  max(Lista2,AcepB,Lista1,GRS_MAX).
```

El predicado `comparable` nos permite comprobar que una palabra está en la lista de sinónimos de otra, una vez fijada determinada acepción de esta última. Así

`comparable(A, B, AcepA)`

se lee: la palabra A es comparable con otra B en determinada acepción AcepA de A.

El predicado determina que dos palabras son comparables, fijada una de las acepciones de la primera, si

- X es la lista de sinónimos de la primera para la acepción AcepA, y
- la segunda es uno de los elementos de X.

```
%COMPARABILIDAD ENTRE PALABRAS
comparable(A,B,AcepA):-
  sin_dic(A,X,AcepA),
  miembro(B,X).
```

El predicado `miembro` es usual en el tratamiento de listas en Prolog, e indica si determinado elemento pertenece o es miembro de una lista:

```
miembro(Elemento, Lista)
```

Un elemento pertenece o es miembro de una lista, si
1.- es su cabeza, o
2.- es miembro de la cola.
En Prolog esto puede expresarse así:

```
%PERTENENCIA A UNA LISTA
miembro(A,[A|B]).
miembro(A,[B|C]):-
  miembro(A,C).
```

El máximo de los grados con la correspondiente acepción se calcula mediante el predicado

```
max(Lista_Acepciones, AcepB, Lista_Grados, GRS_MAX)
```

cuyo primer argumento es una lista de acepciones, el segundo la acepción correspondiente al mayor grado, el tercero es una lista de grados y el cuarto el mayor de los grados. El predicado opera de la siguiente manera:
1.- El máximo de una lista con un solo elemento es ese elemento tanto para las acepciones como para los grados.
2.- Si las listas tienen más de un elemento, compruébese si el primer elemento de la lista de los grados es mayor o igual al segundo. En caso afirmativo elimínese el segundo de ellos en ambas listas.
3.- Si el primero no es mayor o igual al segundo, elimínese el primero.
4.- Repítase este proceso hasta que sólo quede un elemento en ambas listas.

```
%MÁXIMO DE LOS GRADOS DE UNA LISTA Y ACEPCIÓN
CORRESPONDIENTE A ESE GRADO MÁXIMO
max([B],B,[A],A).
max([D,E|F],Y,[A,B|C],X):-
  A >= B,
  max([D|F],Y,[A|C],X),
  !.
max([D,E|F],Y,[A,B|C],X):-
  max([E|F],Y,[B|C],X).
```

El predicado `sin_acep` hace el cálculo para dos acepciones concretas de dos palabras

```
sin_acep(A, AcepA, B, Acep, GRS)
```

Una palabra A en la acepción AcepA es sinónima de una B en la acepción AcepB en un grado GRS si

- no es el caso que A coincida con B (esto evita que el compilador haga el cálculo del grado de sinonimia de dos palabras iguales dentro de una misma acepción, puesto que el resultado será trivialmente 1)
- X es la lista de los sinónimos de A para la acepción AcepA
- Y es la lista de los sinónimos de B para la acepción AcepB
- U es la unión de X e Y
- I la intersección de X e Y
- NU la cardinalidad de la unión U
- NI la cardinalidad de la intersección I
- el grado de sinonimia GRS es NI/NU, y
- GRS es distinto de 0 (esto evitará que el compilador, dada una palabra, dé como resultado palabras que no son sinónimas de ella, es decir, palabras cuyo grado de sinonimia con la primera sea 0).

```
%SINONIMIA ENTRE ACEPCIONES
sin_acep(A,AcepA,B,AcepB,GRS):-
  not(A=B),
  sin_dic(A,X,AcepA),
  sin_dic(B,Y,AcepB),
  union(X,Y,U),
  inter(X,Y,I),
  card(U,NU),
  card(I,NI),
  GRS is (NI/NU),
  GRS \== 0.
```

Si en lugar de GRS \== 0 se pone GRS >= N (se lee GRS mayor o igual a N), donde N es un número real entre 0 y 1, estaremos fijando un umbral por debajo del cual el compilador no admitirá a las palabras como sinónimas.
El predicado `unión` establece el resultado de la unión de dos listas y posee tres listas como argumentos:

```
unión(Lista1, Lista2, ListaResultado)
```

El predicado se define de la siguiente manera:
1.- La unión de la lista vacía con cualquier lista es la misma lista.
2.- Si la primera lista no es vacía, compruébese que la cabeza no pertenece a la segunda, en caso de ser así, el resultado será la segunda lista con ese elemento añadido.
3.- Si la cabeza de la primera lista pertenece a la segunda lista no se añada dicho elemento.
4.- Repítase el proceso con los demás elementos hasta agotar la primera lista.
El código es el siguiente:

```
%UNIÓN DE LISTAS
union([],X,X).
union([A|B],Y,[A|Z]):-
  not(miembro(A,Y)),
  union(B,Y,Z),
  !.
union([A|B],Y,Z):-
  union(B,Y,Z).
```

El predicado `inter` establece el resultado de la intersección de dos listas y posee igualmente tres listas como argumentos:

```
inter(Lista1, Lista2, ListaResultado)
```

El predicado se define de la siguiente manera:
1.- La intersección de la lista vacía con cualquier lista, resulta ser la lista vacía.
2.- Si la primera lista no es vacía compruébese que la cabeza pertenece a la segunda, en caso de ser así, añádase la cabeza a la lista del resultado.
3.- Si la cabeza de la primera lista no pertenece a la segunda lista no se añada dicho elemento.

4.- Repítase el proceso con los demás elementos hasta agotar la primera lista.
El código es el siguiente:

```
%INTERSECCIÓN DE LISTAS
inter([],X,[]).
inter([A|B],Y,[A|Z]):-
  miembro(A,Y),
  inter(B,Y,Z),
  !.
inter([A|B],Y,Z):-
  inter(B,Y,Z).
```

El predicado `card` indica el número de elementos de una lista. Su primer argumento es una lista y el segundo un número natural que corresponde a la cardinalidad de la lista:
`card(Lista, Cardinalidad)`
El predicado se define de la siguiente manera:
1.- El número de elementos de la lista vacía es 0.
2.- El número de elementos de una lista no vacía se obtiene sumándole 1 al número de elementos de esa lista, pero sin su cabeza.
El código es el siguiente:

```
%NÚMERO DE ELEMENTOS DE UNA LISTA
card([],0).
card([A|B],C):-
  card(B,E),
  C is E + 1.0.
```

2.1 VERBALIZACIÓN DEL GRADO DE SINONIMIA ENTRE PALABRAS

Como se ha visto, el grado de sinonimia se expresa con un número real entre 0 y 1. Pero a un usuario le puede resultar más transparente una verbalización lingüística que un grado. Una manera inmediata y sencilla de hacer esto es utilizando un predicado análogo al anteriormente explicitado, pero que en lugar de proporcionar un grado de sinonimia proporcione una verbalización:
`sino_verb(A, AcepA, B, AcepB, Verbalización)`
donde A y B son palabras, AcepA y AcepB son sus respectivas acepciones y Verb es la etiqueta lingüística que corresponde al grado de sinonimia. La estructura del predicado es la siguiente:
1.- La verbalización del grado de sinonimia es "palabras idénticas" si tanto las palabras como las acepciones coinciden y pertenecen al diccionario.
2.- Dadas dos acepciones concretas de dos palabras, Verb es la verbalización de su grado de sinonimia si:

- GRS es el grado de sinonimia entre ambas palabras para dichas acepciones,
- las palabras no son iguales, y
- ese grado GRS se verbaliza como Verb.

```
%VERBALIZACIÓN DEL GRADO DE SINONIMIA ENTRE
PALABRAS
sino_verb(A,AcepA,A,AcepA,palabras_identicas):-
  sin_dic(A,X,AcepA).
sino_verb(A,AcepA,B,AcepB,Verb):-
  sinonimo(A,AcepA,B,AcepB,GRS),
  not(A=B),
  verbaliz(GRS,Verb).
```

El predicado `verbaliz` asocia una verbalización a determinado grado de sinonimia,
`verbaliz(Grado_de_sinonimia, Verbalización)`
Su primer argumento es un grado (número real entre 0 y 1) y el segundo la palabra que verbaliza dicho grado. El criterio para la distribución de los intervalos de cada verbalización es arbitrario; lo único que se persigue con esto es establecer un procedimiento razonable para que la respuesta del compilador sea más atractiva. Con tal objetivo se propone la siguiente distribución de los intervalos:
1.- Todo grado mayor que 0 y menor o igual a 0.25 se verbaliza con la etiqueta "muy poco sinónimos",
2.- Todo grado mayor que 0.25 y menor o igual a 0.5 con "poco sinónimos",
3.- Todo grado mayor que 0.5 y menor o igual a 0.75 con "bastante sinónimos", y
4.- Todo grado mayor que 0.75 y menor o igual a 1 con "muy sinónimos".
El siguiente código permite hacer esto:

```
%VERBALIZACIÓN DE LOS GRADOS
verbaliz(Grado,muy_poco_sinonimos):-
  Grado > 0,
  Grado =< 0.25.
verbaliz(Grado,poco_sinonimos):-
  Grado > 0.25,
  Grado =< 0.5.
verbaliz(Grado,bastante_sinonimos):-
  Grado > 0.5,
  Grado =< 0.75.
verbaliz(Grado,muy_sinonimos):-
  Grado > 0.75,
  Grado =< 1.0.
```

2.2 EJEMPLOS DE SINONIMIA ENTRE PALABRAS

Podemos ahora preguntar cuál es el grado de sinonimia entre dos palabras pertenecientes al diccionario, como "licencia" y "concesión". Una consulta de ese tipo se formula de la siguiente manera:

```
sinonimo(concesion,Acep_concesion,licencia,Acep_licencia,Grado).
```

Como es usual en Prolog, las palabras en mayúsculas se toman como variables que el programa debe instanciar al responder. El compilador respondería:

```
Acep_concesion = 1
Acep_licencia = 1
Grado = 0.375
```

Asimismo, podremos preguntar qué palabras son sinónimas de "concesión", indicando las acepciones de ambas palabras y el grado de sinonimia entre ellas. Esta consulta se formula así:

```
sinonimo(concesion, Acep_concesion, Y, Acep_Y, Grado).
```

y el compilador daría la siguiente respuesta:

```
Acep_concesion = 1          Acep_concesion = 1
Y = concesion               Y = gracia
Acep_Y = 1                  Acep_Y = 2
Grado = 1.0                 Grado = 0.142857
```

```
Acep_concesion = 2
Y = concesion
Acep_Y = 2
Grado = 1.0

Acep_concesion = 1
Y = permiso
Acep_Y = 1
Grado = 0.2

Acep_concesion = 1
Y = licencia
Acep_Y = 1
Grado = 0.375

Acep_concesion = 1
Y = privilegio
Acep_Y = 1
Grado = 0.333333

Acep_concesion = 2
Y = epitrope
Acep_Y = 2
Grado = 1
```

Con el predicado `sino_verb` podemos hacer la primera de las consultas anteriores:

```
sino_verb(concesion,Acep_concesion,licencia,Acep_licencia,Verb).
```

obteniendo la verbalización del grado de sinonimia de ambas palabras:

```
Acep_concesion = 1
Acep_licencia = 1
Verb = poco_sinonimos
```

3 SINONIMIA ENTRE ORACIONES

Si concebimos la sinonimia como una cuestión de grado, ¿cómo afecta al grado de sinonimia de dos oraciones la sustitución de una palabra por su sinónima?. En lo sucesivo intentaremos establecer una serie de propuestas que den cuenta de esta cuestión.

Dos oraciones son sinónimas con un grado de sinonimia GRS si sus palabras coinciden o son sinónimas en ese grado respectivamente. Cuando dos oraciones difieren en dos o más palabras y estas son sinónimas, para calcular el grado de sinonimia de ambas oraciones utilizaremos el producto de los grados de sinonimia de esas palabras respectivamente.

Considerando a una oración como una lista, tal y como es usual en el tratamiento del lenguaje natural mediante Prolog, el predicado `orac_sin` expresa la sinonimia entre oraciones:

```
orac_sin(Oración1, Oración2, Grado_de_sinonimia, Acep1, Acep2)
```

Los dos primeros argumentos son listas donde se incluirán oraciones del lenguaje natural, el tercero es el grado de sinonimia entre ambas oraciones, el cuarto es una lista cuyos elementos serán las palabras sustituidas en la primera oración seguidas de la acepción correspondiente y el quinto lo mismo que el cuarto pero para la segunda oración. El predicado tiene la siguiente estructura:

1.- La lista vacía es sinónima de la lista vacía en grado 1.

2.- Si las cabezas son iguales respectivamente en las listas que representan la primera y segunda oración, entonces el grado de sinonimia no variará.

3.- Si las cabezas no son iguales, entonces:

- si son sinónimas, en la penúltima lista se incluirá la cabeza de la primera oración seguida de la acepción que la hace sinónima de la cabeza de la segunda; en la última lista se incluirá la cabeza de la segunda oración seguida de la acepción que la hace sinónima de la primera.
- multiplíquese el grado de sinonimia que la oración tenga en ese momento por el grado de sinonimia de las cabezas.

4.- Repítase el procedimiento anterior con todas las palabras de las oraciones.

```
%SINONIMIA ENTRE ORACIONES
orac_sin([],[],1.0,[],[]).
orac_sin([A|B],[A|D],X,Y,Z):-
  orac_sin(B,D,X,Y,Z).
orac_sin([A|B],[C|D],Y,[A,Acep_A|W], [C,Acep_C|Q]):-
  sinonimo(A,Acep_A,C,Acep_C,Z),
  orac_sin(B,D,X,W,Q),
  Y is X*Z.
```

3.1 SINONIMIA ENTRE ORACIONES FIJANDO LAS ACEPCIONES

Cuando empleamos palabras en una oración usualmente lo hacemos con una acepción concreta (salvo que estemos haciendo un uso del lenguaje distinto del común, por ejemplo un uso literario). Aunque no siempre es el caso, la oración suele darnos pistas sobre cuál de las acepciones de determinada palabra se está empleando en ella. Otras veces tendremos que recurrir a otro tipo de información extralingüística para dilucidar este problema.

Pueden proponerse algunos criterios para detectar cuál de las acepciones de una palabra es la empleada en una oración. Dichos criterios no deben ser demasiado estrictos porque no son aplicables a la totalidad de los casos. Algunos de ellos pueden ser los siguientes:

1.- Si en una misma oración aparecen dos palabras que son sinónimas respectivamente en determinadas acepciones, es muy probable que dichas acepciones sean las utilizadas en la oración.

2.- Muchas veces, determinadas palabras, que no tienen porque ser sinónimas de una dada, son indicio de que dicha palabra se está empleando en determinada acepción. Por ejemplo, la palabra "banco" tiene en castellano varias acepciones, como "entidad financiera", "asiento para varias personas" o "grupo extenso de peces"; pero si en determinada oración aparece junto a palabras como "dinero", "cheque", "crédito", "préstamo", etc. será un buen indicio de que "banco" se está empleando en la primera de las acepciones.

3.1.1 Primer criterio para fijar las acepciones

Para dar cuenta del primero de los criterios utilizaremos el siguiente predicado:

```
orac_sinónima(Oración1,Oración2,Grado_de_sinonimia,Acep1,Acep2)
```

Este predicado está formado por los mismos argumentos que `orac_sin`, pero tiene en cuenta que la existencia de determinadas palabras en la oración pueden fijar las acepciones de otras. Su estructura es la siguiente:

1.- Una oración A es sinónima de otra B (fijando las acepciones) si:

- Z es la lista de acepciones de palabras fijadas en A,
- A es sinónima de B, y

• ni Z ni la lista de acepciones de B son vacías (esto evita que el programa proporcione como respuesta el caso trivial de que dos oraciones idénticas son sinónimas en grado 1 con las listas de acepciones de A y B vacías).

2.- Cuando en la oración no hay ninguna palabra que fije las acepciones de ninguna otra, entonces la lista de acepciones de A es vacía; en este caso la respuesta del sistema debe combinar todas las acepciones de las palabras sin fijar ninguna.

El código es el siguiente:

```
%SINONIMIA ENTRE ORACIONES FIJANDO LAS
ACEPCIONES
orac_sinonima(A,B,GR,Z,Y):-
  acep_orac_orac(A,A,Z),
  orac_sin(A,B,GR,Z,Y),
  not(Z = []),
  not(Y = []).
orac_sinonima(A,B,GR,W,Y):-
  acep_orac_orac(A,A,Z),
  Z = [],
  orac_sin(A,B,GR,W,Y),
  not(W = []),
  not(Y = []).
```

El predicado `acep_orac_orac` es el que nos permite comprobar si se pueden asociar todas las palabras de una oración con todas las palabras de otra. Eventualmente, en nuestro caso, esta segunda oración será la misma que la primera, aunque pudiera no ser así si lo que queremos es fijar las acepciones de las palabras de una oración con respecto a otras oraciones distintas a la dada. El predicado posee tres listas como argumentos:

`acep_orac_orac(Oración,Oración,Lista_de_acepciones)`

En las dos primeras listas se incluirán oraciones (eventualmente la misma) y en la tercera palabras seguidas de las acepciones fijadas.

1.- Si la primera lista es vacía, la lista de acepciones también lo es.

2.- Si la primera lista no es vacía,

• fíjense las acepciones de la cabeza de la primera lista con las palabras de la segunda,

• verifíquese que el resultado del punto anterior no sea la lista vacía (es decir, que ninguna de las palabras de la segunda lista se puedan asociar con la cabeza de la primera); y

• elimínense las acepciones que se repiten,

• inclúyase en la lista de acepciones la cabeza de la primera lista seguida de la acepción fijada.

3.- En caso de no cumplirse 2, si la primera lista no es vacía,

• fíjense las acepciones de la cabeza de la primera lista con las palabras de la segunda,

• verifíquese que el resultado del punto anterior no sea la lista vacía (es decir, que ninguna de las palabras de la segunda lista se puedan asociar con la cabeza de la primera); y

• elimínense las acepciones que se repiten,

• si el resultado asocia varias acepciones, inclúyase en la lista de acepciones la cabeza de la primera lista seguida de una variable (con lo cual el programa proporcionará todas las combinaciones de acepciones, ya que no hay forma de decidir cual de las acepciones asociadas es la empleada en la oración),

4.- Repítase el proceso con las demás palabras hasta agotar la primera lista.

El código es el siguiente:

```
%FIJANDO LAS ACEPCIONES DE UNA LISTA CON OTRA
acep_orac_orac([],X,[]).
acep_orac_orac([A|B],X,[R,S|Y]):-
  acep_pal_orac(A,X,Z),
  not(Z=[]),
  quitar_rep(Z,[R,S]),
  acep_orac_orac(B,X,Y),
  !.
acep_orac_orac([A|B],X,[A,K|Y]):-
  acep_pal_orac(A,X,Z),
  not(Z=[]),
  quitar_rep(Z,W),
  not(num(W,2)),
  acep_orac_orac(B,X,Y),
  !.
acep_orac_orac([A|B],X,Y):-
  acep_orac_orac(B,X,Y).
```

El predicado `acep_pal_orac` es el que, dada una palabra y una oración, proporciona una lista de las acepciones de la palabra que resultan de asociarla con las palabras de la oración. En el resultado aparece la palabra inicial seguida de la acepción empleada en la asociación, repitiéndose esto tantas veces como asociaciones se hayan realizado. Sus argumentos son una palabra, una lista donde incluiremos una oración y una lista donde se incluirá la palabra con las acepciones que resultan de las asociaciones realizadas.

`acep_pal_orac(Palabra,Oración,Lista_de_acepciones)`

La estructura del predicado es la siguiente:

1.- Si tenemos una palabra y la lista vacía, la lista de las acepciones será la lista vacía.

2.- Si tenemos una palabra y una lista no vacía, añádase a la lista de acepciones la palabra A seguida de la acepción correspondiente AcepA si,

• A y la cabeza de la lista son distintas, y

• A y la cabeza de la lista son asociables para AcepA.

3.- Repítase el proceso hasta agotar la lista.

```
%FIJANDO LAS ACEPCIONES DE UNA PALABRA CON UNA
LISTA
acep_pal_orac(A,[],[]).
acep_pal_orac(A,[B|C],[A,AcepA|E]):-
  not(A=B),
  asocia(A,B,AcepA),
  acep_pal_orac(A,C,E),
  !.
acep_pal_orac(A,[B|C],D):-
  acep_pal_orac(A,C,D).
```

Cuando en la oración hay varias palabras asociables con la primera para la misma acepción, en la última lista aparecerá la acepción tantas veces como asociaciones se

hayan hecho. Para evitar la repetición de acepciones, utilizaremos el predicado quitar_rep, cuyos dos argumentos son listas:

```
quitar_rep(Lista_con_repetición,Lista_sin_repetición)
```

El primer argumento es la lista con acepciones repetidas y el segundo la lista sin repetición de acepciones. La estructura es la siguiente:

1.- La eliminación de las repeticiones en la lista vacía es la lista vacía.

2.- Si la lista no es vacía elimínense los dos primeros elementos si coinciden respectivamente con los dos siguientes.

3.- Repítase el proceso anterior chequeando los elementos de la primera lista dos a dos hasta agotarla.

```
%ELIMINACIÓN DE ACEPCIONES REPETIDAS
quitar_rep([],[]).
quitar_rep([A,AcepA,B,AcepB|C],[B,AcepB|D]):-
  [A,AcepA]=[B,AcepB],
  quitar_rep([B,AcepB|C],[B,AcepB|D]),
  !.
quitar_rep([A,AcepA|C],[A,AcepA|D]):-
  quitar_rep(C,D).
```

El predicado asocia empleado en acep_pal_orac nos permite asociar palabras. Este predicado está formado por dos palabras y una acepción correspondiente a la primera palabra:

```
asocia(A, B, Acep_A)
```

Asociaremos dos palabras:

- cuando la primera en determinada acepción sea sinónima de la segunda en cualquier acepción en cierto grado, o
- la segunda en cualquier acepción sea sinónima de la primera en determinada acepción en cierto grado.

```
%ASOCIACIÓN DE PALABRAS PARA EL PRIMER CRITERIO
asocia(A,B,AcepA):-
  sinonimo(A,AcepA,B,AcepB,GR);
  sinonimo(B,AcepB,A,AcepA,GR).
```

3.1.2 Segundo criterio para fijar las acepciones

Para dar cuenta del segundo de los criterios que fijan las acepciones en una oración, se necesita crear otra base de datos que sea capaz de asociar determinadas palabras con otras y fijar las acepciones de estas últimas. Una opción podría ser considerar que verbos como "tener" indican que se está utilizando la primera acepción de "concesión" y verbos como "emplear" indican que se usa la segunda. Luego podríamos crear una nueva base de datos donde apareciesen predicados como el siguiente:

```
%ASOCIACIÓN DE PALABRAS PARA EL SEGUNDO CRITERIO
asocia(concesion,tengo,1).
asocia(concesion,empleo,2).
...
```

Es evidente que sería válida toda conjugación del verbo "tener" y "emplear" en cualquier tiempo, modo, número y persona. Los predicados, quitar_rep, acep_pal_orac, acep_orac_orac y orac_sinónima son análogos a los definidos para el criterio anterior. El predicado asocia definido allí debe ser sustituido por la base de datos de arriba.

3.2 VERBALIZACIÓN DEL GRADO DE SINONIMIA ENTRE ORACIONES

Tal y como se ha hecho para la sinonimia entre palabras podemos definir un predicado que en lugar de proporcionar el grado numérico de sinonimia, dé una verbalización del mismo. Es el siguiente:

```
orac_sinónima_verb(Oración1,Oración2,Verbalización,Acep1,Acep2)
```

Este predicado está formado por dos oraciones, la verbalización del grado de sinonimia entre ellas y dos listas que contienen las acepciones fijadas en la primera y segunda oración respectivamente. Opera de la siguiente manera:

Verb es la verbalización del grado de sinonimia de dos oraciones si:

- GRS es el grado de sinonimia entre ambas, y
- GRS se verbaliza como Verb.

El código es el siguiente:

```
%VERBALIZACIÓN DEL GRADO DE SINONIMIA ENTRE
ORACIONES
orac_sinonima_verb(A,B,Verb,W,Y):-
  orac_sinonima(A,B,GRS,W,Y),
  verbaliz(GRS,Verb).
```

3.3 EJEMPLOS DE SINONIMIA ENTRE ORACIONES

Podemos preguntar el grado de sinonimia de dos oraciones:

```
orac_sinonima([el,chico,tiene,licencia],[el,chic
o,tiene,concesion],GR,Y,Z).
```

a lo que el compilador contestará:

```
GR = 0.375
Y = [licencia,1]
Z = [concesion,1]
```

Si preguntamos ahora:

```
orac_sinonima([el,que,tiene,licencia,tiene,conce
sion],B,Grado,X,Y).
```

veremos que el sistema fija la acepción 1 de "licencia" y la acepción 1 de "concesión" debido a que ambas palabras aparecen en la oración y son asociables en esas acepciones:

```
B = [el,que,tiene,licencia,tiene,concesion]
Grado = 1
X = [licencia,1,concesion,1]
Y = [licencia,1,concesion,1]

B = [el,que,tiene,licencia,tiene,permiso]
Grado = 0.2
X = [licencia,1,concesion,1]
Y = [licencia,1,permiso,1]

B = [el,que,tiene,licencia,tiene,licencia]
Grado = 0.375
X = [licencia,1,concesion,1]
Y = [licencia,1,licencia,1]

...
```

```
B = [el,que,tiene,permiso,tiene,concesion]
Grado = 0.625
X = [licencia,1,concesion,1]
Y = [permiso,1,concesion,1]

...
```

Utilizando ahora el código definido para el segundo criterio podemos hacer la siguiente consulta:

```
orac_sinonima([yo,tengo,concesion],B,Grado,X,Y).
```

con lo que podemos observar como el sistema fija la acepción 1 de "concesión":

```
B = [yo,tengo,concesion]
Grado = 1
X = [concesion,1]
Y = [concesion,1]

B = [yo,tengo,gracia]
Grado = 0.142857
X = [concesion,1]
Y = [gracia,2]

B = [yo,tengo,permiso]
Grado = 0.2
X = [concesion,1]
Y = [permiso,1]

B = [yo,tengo,privilegio]
Grado = 0.333333
X = [concesion,1]
Y = [privilegio,1]

B = [yo,tengo,licencia]
Grado = 0.375
X = [concesion,1]
Y = [licencia,1]
```

Si probamos lo mismo con la oración "empleo una concesión en el texto":

```
orac_sinonima([empleo,una,concesion,en,el,texto]
,B,Grado,X,Y).
```

el compilador fijará la acepción 2 de "concesión":

```
B = [empleo,una,concesion,en,el,texto]
Grado = 1
X = [concesion,2]
Y = [concesion,2]

B = [empleo,una,epitrope,en,el,texto]
Grado = 1
X = [concesion,2]
Y = [epitrope,2]
```

Con el predicado `orac_sinónima_verb` podemos hacer la primera consulta:

```
orac_sinonima_verb([el,chico,tiene,licencia],[el
,chico,tiene,concesion],Verb,Y,Z).
```

obteniendo como respuesta:

```
Verb = poco_sinonimos
Y = [licencia,1]
Z = [concesion,1]
```

4 CONCLUSIONES

Se ha propuesto un programa Prolog que analiza el grado de sinonimia entre las palabras de un diccionario de sinónimos y como la sustitución de dos palabras sinónimas en una oración puede afectar al grado de sinonimia entre la oración original y aquella en la que se ha sustituido la palabra.

Como sucede frecuentemente cuando se trata con problemas relativos al lenguaje natural, las cuestiones tienen un dominio tan extenso que pueden provocar fluctuaciones respecto a los resultados ofrecidos aquí. Se pueden usar otras funciones para medir el grado de sinonimia, fijar un umbral por debajo del cual no se discrimine o adoptar otros criterios de verbalización. En cualquier caso, la sinonimia es un tema que se ha estudiado relativamente poco y este trabajo ha pretendido elucidar sólo algunos aspectos relativos a su procesamiento computacional.

Referencias

[1] CHIERCHIA, G.; McCONNELL-GINET (1990) *Meaning and Grammar. An Introduction to Semantics*, MIT Press.

[2] COVINGTON, M. A. (1994) *Natural Language Processing for Prolog Programmers*, Prentice-Hall.

[3] CRUSE, D. A. (1997) *Lexical Semantics*, Cambridge U. P.

[4] GILI GAYA, S.(1997) *Diccionario Avanzado de Sinónimos y Antónimos de la Lengua Española*, VOX (reimpresión octubre 1997).

[5] SPARCK JONES, K. (1986) *Synonymy and Semantic Classification*, Edinbürgh U. P.

[6] TRILLAS, E. (1993) Apunte sobre la Indistinguibilidad; *Theoria*, Vol. VIII, N° 19, pp. 23-49.

[7] TRILLAS, E.; LÓPEZ DE MÁNTARAS, R. (1984) Towards a Measure of the Degree of Synonymy; SÁNCHEZ, E. (ed.) *Fuzzy Information, Knowledge Representation and Decision Analysis*, Pergamon Press.