Hacia un tratamiento computacional de la sinonimia

Santiago Fernández Lanza
sflanza@usc.es

Alejandro Sobrino Cerdeiriña
lflgalex@usc.es

Resumen

Este trabajo tiene tres partes. En la primera se apuntan algunos pasajes históricos vinculados al desarrollo de la sinonimia y se distinguen dos formas de considerarla: como una relación imprecisa entre palabras, como muestran los diccionarios de sinónimos, o como una relación precisa entre oraciones, como ocurre en la transformación de pasiva. En la segunda parte se exponen brevemente algunos intentos de automatizar la sinonimia llevados a cabo en el ámbito del aprendizaje asistido por ordenador, la clasificación mecánica de datos o la recuperación de información en Internet. En el último apartado se exponen dos programas para automatizar la sinonimia; proponiendo los primeros pasos en la automatización de un diccionario de sinónimos del español y modelando la transformación de pasiva. En ambos casos se usan técnicas de programación lógica. El trabajo termina con unas breves indicaciones sobre la posible proyección de esta iniciativa y con la bibliografía.

1. Introducción

La preocupación por la sinonimia, como relación de semejanza significativa, es antigua [5]. Aristóteles propuso ya en los *Tópicos* (I 7:103a6-32) una aproximación a este concepto: *"Antes de nada hay que precisar, acerca de lo idéntico, de cuantas maneras se dice".* Con su descripción, inauguró la preocupación por este tema e influyó en el propósito de agrupar como palabras sinónimas a aquellas que coincidiendo significativamente, presentan no obstante algunas diferencias. En Roma se recogió esta tradición. Seleuco escribió un tratado *Sobre la diferencia en los sinónimos* y Amonio presentó un léxico en *Sobre expresiones semejantes y diferentes.* Griegos y romanos apuntaron a dos características fundamentales de la sinonimia: 1) es una característica del significado de las palabras y atiende a la pluralidad de significantes de un mismo referente. 2) La sinonimia es una relación de semejanza de significados. Si bien los intentos de definir o caracterizar a la sinonimia son antiguos, su estudio sistemático como relación léxica es más reciente. Nace en Francia, a principios del siglo XVIII con la obra de Gabriel Girard *La justesse de la langue françoise, ou les differentes significations des mots qui passent pour synonymes* (1718), que puede considerarse como el primer diccionario de sinónimos.

Los diccionarios de sinónimos muestran a la sinonimia como una característica léxica y aproximada. Como tal, a veces sigue reglas léxicas, como acontece cuando se prefija una palabra de diferentes maneras. P. ej., se obtienen sinónimos en algún grado con la composición por concatenación de alguno de los siguientes prefijos *{pre, post, ultra, super, sub}* con la palabra *código*. Pero, aunque la sinonimia de palabras es la más habitual, no es la única. También hay sinonimia de oraciones.

La sinonimia de oraciones responde a reglas sintácticas -siguiendo a Chomsky, a reglas transformacionales-, y recoge el aspecto preciso de esta relación. El caso de la transformación de pasiva es un claro ejemplo de conversión de una oración en otra que deja el significado inalterado.

Las características imprecisas de la sinonimia del lenguaje natural serán estudiadas en este trabajo en términos de la automatización de un diccionario de sinónimos. Las precisas, como la modelización de la transformación de pasiva. Ambas tareas serán realizadas en el ámbito del procesamiento del lenguaje natural y ocuparan el punto 3 de este estudio. Antes, en 2, repasaremos algunos intentos de automatizar la sinonimia.

2. Algunos intentos de automatizar la sinonimia

2.1. En aplicaciones a sistemas de aprendizaje asistido por ordenador

E. Foxley y M. Gwei [4] diseñaron un programa para la generación automática de sinónimos utilizando como referencia el Roget's Thesaurus, un diccionario conceptual del inglés. Su objetivo es proporcionar una especie de algoritmo para encontrar, de modo mecánico, la palabra que mejor exprese un concepto. Se resume en los siguientes pasos: 1) Buscar en el índice del diccionario la palabra que más se asimila al concepto y anotar los parágrafos en los que aparece así como su categoría gramatical. 2) Comparar el significado del concepto fuente con las palabras halladas. 3) Escoger aquella que mejor lo expresa. 4) Si la palabra seleccionada es satisfactoria, detener la búsqueda; el parágrafo correspondiente contendrá la palabra adecuada y sus sinónimos. En caso contrario, volver al paso 2.

En este trabajo, la generación automática de sinónimos atiende fundamentalmente a la manipulación de afijos como heurística principal para derivar palabras sinónimas y la decisión sobre qué palabra es la más adecuada para expresar un concepto o idea se hace considerando medidas de confianza (que cuantifican la certidumbre de que la palabra exprese el concepto) y medidas de relevancia (que pondera la relevancia de un parágrafo respecto a todos los que se han recorrido para encontrar la palabra adecuada).

Esta automatización debe ayudar a escoger la palabra más oportuna cuando se tiene una idea de lo que se quiere decir, pero no se encuentra el léxico adecuado. Tendrá, por tanto, utilidad en el ámbito de los interfaces conversacionales, asistiendo, de una manera mecánica y eficiente, en la búsqueda de sinónimos.

2.2. En clasificación automática

B. Cases [1] propone automatizar un diccionario de sinónimos utilizando como modelo un autómata Q-diam que se automodifica. El proceso de automodificación lleva a evitar las paradojas que se producen de los diccionarios manuales cuando se hace una búsqueda cruzada y reversible, en la que pueden aparecer una palabra y su negación. P. ej., si se interpreta que *violeta* y *amarillo* son colores opuestos (de hecho, son complementarios en la escala cromática) resulta que, en un diccionario de sinónimos del español, *blanco=amarillento*, *blanco=cárdeno*, pero *amarillento=no(cárdeno)*. El objetivo es automatizar un diccionario de manera que una palabra afirmada y negada no pertenezcan a la misma clase. Al identificar precisión con ausencia de contradictoriedad se obtienen, en opinión de la autora, clases de equivalencia de palabras sinónimas, que se pueden intercambiar en cualquier contexto sin pérdida de significado. Para evitar las paradojas se emplea una lógica tetra-valuada con valores $I \leq T = F \leq J$, donde T y F son valores booleanos, I es el valor e inicio y J es el valor paradójico, que representa la ecuación autonegada $X = no(X)$. Las transiciones se calculan con un autómata Q-diam, donde cada grupo de sinónimos está representado por un hipergrafo o célula, los valores lógicos son los posibles estados y la transición entre estados se calculan de modo que de una célula no se puedan hacer transiciones a una que tenga valor T y a otra que tenga valor F, ya que en ese caso se alcanzaría el valor paradójico. En este recorrido del autómata se alcanzan estados finales en los que habrá grupos de palabras equivalentes, en el sentido que no son contradictorias y, por tanto, que son sustituibles en cualquier contexto.

2.3. En recuperación de información en Internet

S. J. Green [7] presenta un sistema de generación de hipertexto basado en el encadenamiento léxico, que justifica establecer enlaces no sólo entre palabras que son iguales, sino también entre palabras que tienen alguna vecindad significativa -p. ej., si son sinónimas. El autor toma, en este caso, los encadenamientos léxicos de WordNet, que muestran tres tipos de relaciones entre palabras: a) extrafuerte, si se repite la misma palabra; b) fuerte, si dos palabras están en el mismo *synset* o unidas por la relación de antonimia, *es-un* o *inluido-en* y c) intermedia si hay un camino admisible en el grafo WordNet entre los *synsets* que contienen a las palabras consideradas. Utilizando estas relaciones se pueden justificar y ponderar enlances intradocumentales e interdocumentales.

La ponderación de los enlaces intradocumentales resulta de hallar la similaridad de los vectores de densidad (fracción de palabras de un parágrafo que están presentes en un enlace) de los dos o más parágrafos que se pretender enlazar. De ello resulta una matriz simétrica $k \times k$ (k es el número de parágrafos), a partir del cual es posible hallar las desviaciones media y estándar de las similaridades halladas. Con esos datos, si dos parágrafos son más similares que un umbral dado, entonces se determina establecer un enlace entre ellos. La ponderación de los enlaces interdocumentales se hace relativa a dos vectores. El vector 1 contiene una ponderación del número de ocurrencias de un *synset* particular en las palabras que conforman las oraciones de un documento. El vector 2 contiene una ponderación del número de ocurrencias de un *synset* (diferente del presente en el primer vector) relacionado, por ej., por antonimia, en las palabras que conforman las frases de un documento. Dados dos documentos, D1 y D2, se calculan tres similaridades: a) La similaridad de los vectores 1 de D1 y D2; b) la similaridad del vector 1 de D1 con el vector 2 de D2; La similaridad del vector 2 de D1 con el vector 1 de D2. La suma (ponderada) de estas medidas ofrece la similaridad total entre los dos documentos.

Estos tres intentos de automatizar la sinonimia constituyen empresas loables, pero presentan deficiencias. En el primer caso, se plantea la generación automática de un diccionario de sinónimos basándose fundamentalmente en la manipulación de categorías afijales, pero extender ese procedimiento u otros de naturaleza similar al resto del léxico parece difícil. El segundo intento de automatizar un diccionario es una aproximación muy imaginativa, pero artificial, ya que el propósito es eliminar la similaridad significativa de las palabras que son sinónimas, confeccionando un diccionario de sinónimos perfectos. El tercer y último caso descrito utiliza un diccionario de sinónimos (WordNet) para automatizar búsquedas en la red, pero no dice nada acerca de cómo mecanizarlo.

En el apartado 3 proponemos un intento, todavía inicial, de mecanizar un diccionario de sinónimos con técnicas de Programación lógica (3). Además, se ofrecerá un programa para mecanizar sinonimia oracional, presente p. ej. en la transformación de pasiva.

3. Un acercamiento a la sinonimia desde el procesamiento del lenguaje natural

3.1. La sinonimia como relación aproximada: Sinonimia entre palabras

En este apartado se propone un programa Prolog que calcula el grado de sinonimia entre dos palabras utilizando medidas de similaridad. Los ejemplos pertenecen a [6]. 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 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 [6]:

```
sin_dic(concesion,[concesion,permiso,licenci
a,gracia,privilegio],1).
sin_dic(concesion,[concesion,epitrope],2).
sin_dic(permiso,[permiso,autorizacion,consen
timiento,licencia,venia,beneplacito,aquiesce
ncia],1).
sin_dic(licencia,[licencia,permiso,autorizac
ion,consentimiento,venia,concesion],1).
sin_dic(licencia,[licencia,abuso,osadia,atre
vimiento,desenfreno,libertinaje],2).
sin_dic(gracia,[gracia,beneficio,favor,merce
d,don,regalo],1).
sin_dic(gracia,[gracia,perdon,indulto],2).
sin_dic(gracia,[gracia,benevolencia,amistad,
afabilidad,agrado],3).
sin_dic(gracia,[gracia,garbo,donaire,sal,sal
ero,angel,atractivo,encanto],4).
sin_dic(gracia,[gracia,chiste,agudeza,ocurre
ncia],5).
sin_dic(privilegio,[privilegio,prerrogativa,
concesion],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 GS mediante una de las siguientes medidas de similaridad habituales -la lista no es exhaustiva-: coeficiente jaccard, coeficiente del dado, coeficiente del coseno, coeficiente de similaridad mutua, coeficiente de solapamiento. (A modo de ilustración, sólo se pone el coeficiente *jaccard*):

Coeficiente *Jaccard*

$$GS = \frac{|ListaA \cap ListaB|}{|ListaA \cup ListaB|}$$

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 `sinonimo` calcula el grado de sinonimia entre dos palabras:
`sinonimo(A, AcepA, B, AcepB, GRS, U, MS)`
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 para el umbral U y la medida de similaridad MS. El código es el siguiente:

```
sinonimo(A,X,A,Y,1.0,_,coeficiente_jaccard):-
   sin_dic(A,Z,X),
   X=Y.
sinonimo(A,AcepA,B,AcepB,GRS_MAX,Umbral,coeficiente_jaccard):-
   comparable(A,B,AcepA),
   findall(GRS,sin_acep(A,AcepA,B,Acep,GRS,Umbral,coeficiente_jaccard),Lista1),
   findall(Acep,sin_acep(A,AcepA,B,Acep,GRS,Umbral,coeficiente_jaccard),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:

```
comparable(A,B,AcepA):-
   sin_dic(A,X,AcepA),
   miembro(B,X).
```

El predicado `miembro` es el habitual en Prolog.

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:

```
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, U, MS)`
Este predicado se lee diciendo que una palabra A en la acepción AcepA es sinónima de una B en la acepción AcepB en un grado GRS para el umbral U y la medida de similaridad MS:

```
sin_acep(A,AcepA,B,AcepB,GRS,Umbral,coeficiente_jaccard):-
   not(A=B),
   sin_dic(A,X,AcepA),
   sin_dic(B,Y,AcepB),
   union(X,Y,U),
   inter(X,Y,I),
   num(U,NU),
   num(I,NI),
   GRS is (NI/NU),
   GRS >= Umbral.
```

Los predicados `union`, `interseccion` y `cardinalidad` se definen de la manera habitual en Prolog.

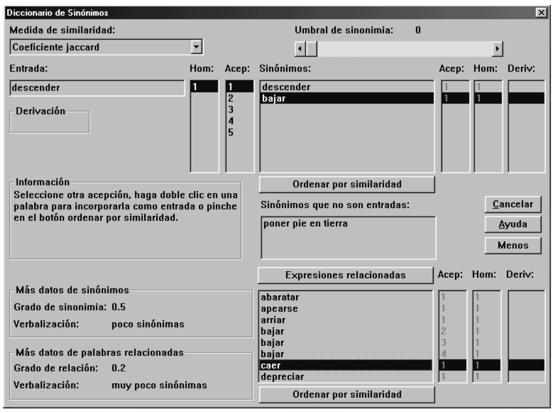
Con este programa es posible preguntar el grado de sinonimia entre dos palabras que pertenezcan al diccionario, tales como “concesión” y “licencia” con umbral 0 y para el coeficiente *jaccard*

```
?- sinonimo(concesion,AcepA,licencia,AcepB,
GRS,0,coeficiente_jaccard).
```

a lo que el compilador responderá:

```
AcepA = 1
AcepB = 1
GRS = 0.375
```

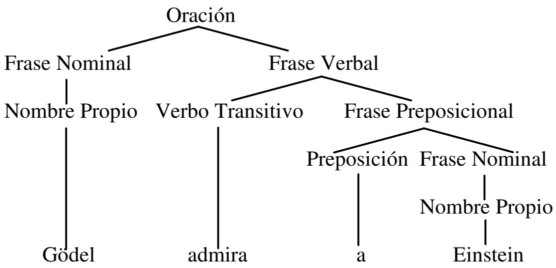
Se dispone de un interfaz gráfico, realizado en Visual Prolog, para consultar los sinónimos de una palabra y sus grados de sinonimia:

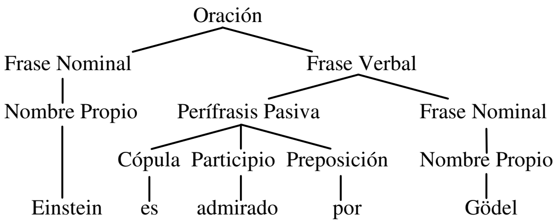


3.2. La sinonimia como relación precisa: la transformación de pasiva

Un caso típico de sinonimia entre dos oraciones es la transformación de pasiva.

Resulta evidente que la oración "Gödel admira a Einstein" tiene el mismo significado que "Einstein es admirado por Gödel", aunque se trata de dos oraciones distintas con una estructura gramatical diferente, como se puede comprobar con sus respectivos árboles gramaticales:





Para detectar de forma automática este tipo de sinonimia se puede recurrir a realizar un análisis semántico con técnicas de Prolog. El resultado debería ser un programa Prolog capaz de fijar el significado de una oración en función del significado de sus componentes y del papel que ejercen estos en su estructura sintáctica, mostrando que aunque "Gödel admira a Einstein" y "Einstein es admirado by Gödel" tienen distinta estructura superficial, responden a la misma estructura profunda, al ser transformable, bajo preservación de verdad, una en la otra. El análisis semántico revelaría que el significado de ambas oraciones es *admira(Gödel,Einstein)*, aunque para significar esto respondan al uso de distintas reglas.

Fieles al paradigma de la programación lógica, el análisis semántico presupone el análisis sintáctico o gramatical, que lo proporciona, para las oraciones antes señaladas, el siguiente programa:

```
oracion(Lista1, Resto):-
   frase_nominal(Lista1, Lista2),
   frase_verbal(Lista2, Resto).
frase_nominal(Lista1, Resto):-
   nombre_propio(Lista1, Resto).
frase_verbal(Lista1, Resto):-
   verbo_transitivo(Lista1, Lista2),
   frase_preposicional(Lista2, Resto).
frase_verbal(Lista1, Resto):-
   perifrasis_pasiva(Lista1, Lista2),
   frase_nominal(Lista2, Resto).
frase_preposicional(Lista1, Resto):-
   preposicion(Lista1, Lista2),
   frase_nominal(Lista2, Resto).
perifrasis_pasiva(Lista1, Resto):-
   copula(Lista1, Lista2),
   participio(Lista2, [por|Resto]).
nombre_propio(['Gödel'|Resto], Resto).
nombre_propio(['Einstein'|Resto], Resto).
verbo_transitivo([admira|Resto], Resto).
preposicion([a|Resto], Resto).
copula([es|Resto], Resto).
participio([admirado|Resto], Resto).
```

A la pregunta de qué oraciones genera:

```
?- oracion(Orac,[]).
```

el compilador responde:

```
...
Orac = [Gödel,admira,a,Einstein]
...
Orac = [Einstein,es,admirado,por,Gödel]
...
```

Una vez que tenemos la gramática que genera las oraciones que deseamos, debemos incorporar en ella los criterios por los que se debe guiar el análisis semántico.

(a) En primer lugar, la manera de representar el significado de un nombre propio es utilizando dicho nombre propio como significado. Esto se realiza en Prolog incorporando la palabra empleada para definir al individuo que denota el nombre propio, como primer argumento del predicado `nombre_propio`. Así tendremos:

```
nombre_propio('Gödel',['Gödel'|Resto],
Resto).
nombre_propio('Einstein',['Einstein'|Resto],
Resto).
```

(b) El significado de un verbo transitivo será una relación, cuyo nombre es la palabra empleada para el verbo y posee dos argumentos: un agente y un paciente. Por tanto, el agente, el paciente y el significado del verbo transitivo serán los tres primeros argumentos del predicado `verbo_transitivo`:

```
verbo_transitivo(Agente,Paciente,admira(Agen
te,Paciente),[admira|Resto], Resto).
```

(c) El significado del participio es el mismo que el de un verbo transitivo:

```
participio(Agente,Paciente,admira(Agente,Pac
iente),[admirado|Resto],Resto).
```

(d) El significado de la perífrasis de pasiva se hereda del significado del participio, que es uno de sus componentes:

```
perifrasis_pasiva(Agente,Paciente,Sgdo,Lista
1,Resto):-
   copula(Lista1,Lista2),
   participio(Agente,Paciente,Sgdo,Lista2,
   [por|Resto]).
```

(e) El significado de una frase nominal es el mismo que el de su único componente; nombre propio:

```
frase_nominal(Sgdo,Lista1,Resto):-
   nombre_propio(Sgdo,Lista1,Resto).
```

(f) El significado de la frase preposicional se hereda del significado de la frase nominal, que es uno de sus componentes:

```
frase_preposicional(Sgdo,Lista1, Resto):-
   preposicion(Lista1, Lista2),
   frase_nominal(Sgdo,Lista2, Resto).
```

(g) La mayor dificultad está en la frase verbal.

Si está en activa, el significado de la frase nominal que va después del verbo transitivo es el paciente. En este caso, como es necesaria la información sobre quién es el agente, que se obtendrá del significado de la frase nominal que encabeza la oración, se incluirá dicha información como primer argumento, mientras que el segundo será el significado de la frase verbal, que resultará de la composición del significado del verbo transitivo con el significado de la frase preposicional que lo sigue:

```
frase_verbal(Agente,Sgdo,Lista1,Resto):-
   verbo_transitivo(Agente,Paciente,Sgdo,
   Lista1,Lista2),
   frase_preposicional(Paciente,Lista2,
   Resto).
```

Si la frase verbal está en pasiva el significado de la frase nominal que va después del verbo transitivo es el agente. En ese caso, como es necesaria la información sobre quién es el paciente, que se obtendrá del significado de la frase nominal que encabeza la oración, se incluirá dicha información como primer argumento, mientras que el segundo será el significado de la frase verbal, que resultará de la composición del significado de la perífrasis pasiva con el significado de la frase nominal que lo sigue:

```
frase_verbal(Paciente,Sgdo,Lista1,Resto):-
   perifrasis_pasiva(Agente,Paciente,Sgdo,
   Lista1,Lista2),
   frase_nominal(Agente,Lista2,Resto).
```

(h) Finalmente, el significado de la oración es el significado de la frase verbal compuesto con el significado de la frase nominal que encabeza la oración, dicha frase nominal proporciona la información de quién es el agente en el caso de las oraciones en activa y quién es el paciente en el caso de las oraciones en pasiva:

```
oracion(Sgdo_orac,Lista1, Resto):-
   frase_nominal(Sgdo_fn,Lista1, Lista2),
   frase_verbal(Sgdo_fn,Sgdo_orac,Lista2,
   Resto).
```

Los demás componentes de la oración no influyen en el significado tal y como aquí es concebido para este ejemplo concreto; sin embargo, son necesarios para la generación de las oraciones gramaticales.

Un predicado que sea capaz de detectar si dos oraciones son sinónimas podría ser entonces el que tiene en cuenta que ambas oraciones son gramaticales y su significado es el mismo. Tal predicado tendrá dos oraciones como argumentos. Veámoslo:

```
sinonima(Orac1,Orac2):-
      oracion(M,Orac1,[]),
      oracion(M,Orac2,[]).
```

Ahora podremos preguntar al compilador cual es el significado de cada una de las oraciones generadas por la gramática:

```
?- oracion(Sgdo,Orac,[]).
...
Sgdo = admira(Gödel,Einstein)
Orac = [Gödel,admira,a,Einstein]
...
Sgdo = admira(Gödel,Einstein)
Orac = [Einstein,es,admirado,por,Gödel]
...
```

También podemos preguntar cuáles de las oraciones generadas por la gramática son sinónimas (evitando oraciones idénticas):

```
?.- sinonima(Orac1,Orac2),Orac1\==Orac2.
...
Orac1 = [Gödel,admira,a,Einstein]
Orac2 = [Einstein,es,admirado,por,Gödel]
...
```

4. Recapitulación final

Se han presentado dos programas Prolog; uno para calcular el grado de sinonimia en un diccionario de sinónimos y otro para hallar oraciones sinónimas cuando una de ellas es la transformación de pasiva de la otra. Ambos programas ilustran como procesar características imprecisas y precisas de la sinonimia y quieren mostrar los beneficios de la utilización de las herramientas formales propias de un lenguaje de programación como Prolog aplicadas al procesamiento del lenguaje natural.

Su implementación puede ser relevante en distintas áreas de la Inteligencia Artificial, como en el desarrollo de entornos de aprendizaje automático flexibles (p. ej., en un sistema que detecte como sinónimas a las siguientes respuestas: *El problema P tiene solución A*, *A es la solución del problema P*, estando predefinida sólo una de ellas) o en el diseño de interfaces de comunicación de sistemas expertos en los que se valore la matización y la flexibilidad que la sinonimia léxica (p. ej., en sistemas informáticos que detecten como sinónimos a *borrar*, *cortar* o *suprimir*).

Agradecimientos

Financiación a cargo del proyecto PGIDT99PXI10502B de la Xunta de Galicia.

Bibliografía

[1] CASES, B., (1996), 'From Synonymy to self-modifyng automata: Q-Diam Language', in Dassow, J. et al. (eds.), *Developments in Language Theory II. At the Crossroads of Mathematics, Computer Science and Biology*. World Scientific, Singapore. pp. 454-459.

[2] CARNAP, R. (1955), 'Meaning and Synonymy in Natural Languages', Philosophical Studies, 7, 33-47.

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

[4] FOXLEY, E. & GWEI, G. M., (1989), 'Synonymy and Contextual Disambiguation of Words', *International Journal of Lexicography*, Vol. 2, nº 2, 111-131.

[5] GARCIA-HERNANDEZ, B., (1997), 'Sinonimia y diferencia de significado', *Revista Española de Lingüística*, 27, 1, 1-31.

[6] GILI GAYA, S. (1997), *Diccionario avanzado de sinónimos y antónimos de la lengua española*, VOX (reimpresión octubre 1997)

[7] GREEN, S. J., (1998) 'Automated link generation: Can we do better than term repetition?, *Computer Networks and ISDN Systems* 30 (1998), 75-84.