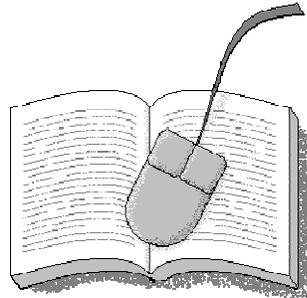


Instituto Politécnico Nacional
Centro de Investigación en Computación
Maestría en Ciencias de la Computación
Laboratorio de Lenguaje Natural y Procesamiento de Texto



**Selección automática
de primitivas semánticas
para un diccionario explicativo
del idioma español**

TESIS QUE PRESENTA

Lic. GABRIELA RIVERA LOZA

PARA OBTENER EL GRADO DE

MAESTRO EN CIENCIAS DE LA COMPUTACIÓN

DIRECTOR DE TESIS

DR. ALEXANDER GELBUKH

CO-DIRECTOR

DR. GRIGORI SIDOROV

México, D.F., 2003

ÍNDICE GENERAL

Resumen	5
Abstract	6
1 Introducción.....	7
2 Métodos de procesamiento automático de información semántica.....	13
3 Selección automática de primitivas semánticas	45
4 Metodología experimental.....	89
5 Conclusiones	99
6 Referencias.....	103
Anexo 1. Código fuente de las funciones principales	108
Anexo 2. Formato del grafo de entrada	133
Anexo 3. Lista de las primitivas encontradas	140
Anexo 4. Lista de los ciclos encontrados	157

ÍNDICE DETALLADO

Resumen	5
Abstract	6
1 Introducción.....	7
<i>1.1 Objetivo principal</i>	<i>8</i>
<i>1.2 Metas particulares</i>	<i>8</i>
<i>1.3 Diccionarios computacionales</i>	<i>9</i>
2 Métodos de procesamiento automático de información semántica.....	13
<i>2.1 Teorías semánticas modernas.....</i>	<i>13</i>
<i>2.2 Teoría clásica del significado.....</i>	<i>14</i>
<i>2.3 Aproximación prototípica</i>	<i>20</i>
<i>2.4 Aproximación relacional</i>	<i>21</i>
2.4.1 Polisemia en la aproximación relacional	21
<i>2.5 Problemas relacionados con polisemia.....</i>	<i>21</i>
2.5.1 Polisemia desde el punto de vista computacional	21
<i>2.6 Procesamiento automático de los diccionarios explicativos.....</i>	<i>26</i>
2.6.1 Aproximaciones basadas en diccionarios.....	27
<i>2.7 Recuperación de información con los métodos de lingüística computacional.....</i>	<i>32</i>
3 Selección automática de primitivas semánticas	45
<i>3.1 Conceptos básicos.....</i>	<i>45</i>
<i>3.2 Algoritmo principal.....</i>	<i>48</i>
<i>3.3 Modelo matemático.....</i>	<i>50</i>
<i>3.4 Implementación del algoritmo</i>	<i>56</i>
3.4.1 Módulo graph	57
3.4.2 Módulo circle	60
3.4.3 Módulo vis	61
3.4.4 Librería circle.....	62
3.4.5 Explicación del código.....	62

3.4.5.1	Módulo <code>circle</code>	79
3.4.5.2	Módulo <code>vis</code>	82
3.4.5.3	Librería <code>circle</code>	84
4	Metodología experimental	89
4.1	<i>Aplicación del algoritmo</i>	<i>¡Error! Marcador no definido.</i>
5	Conclusiones	99
5.1	<i>Herramienta lexicográfica</i>	<i>¡Error! Marcador no definido.</i>
5.2	<i>Trabajo futuro</i>	<i>101</i>
6	Referencias	103
Anexo 1. Código fuente de las funciones principales		108
6.1	Módulo <code>graph.cpp</code>	<i>108</i>
6.2	Módulo <code>circle.cpp</code>	<i>117</i>
6.3	Módulo <code>vis.cpp</code>	<i>122</i>
6.4	Declaraciones: <code>circle.h</code>	<i>130</i>
Anexo 2. Formato del grafo de entrada		133
Anexo 3. Lista de las primitivas encontradas		140
Anexo 4. Lista de los ciclos encontrados		157

LISTA DE FIGURAS

Figura 1. El uso de notaciones lexicográficas para distinguir diferentes sentidos de una palabra.	15
Figura 2: Parte de una red semántica.	28
Figura 3: Representación del significado de la palabra "planta".	33
Figura 4: Modelo de Miller y Johnson Laird.	34
Figura 5: Relación tipo-subtipo.	42
Figura 6. Uno de los sentidos de la palabra "prolepsis".	49
Figura 7. Definición de "cobla".	50
Figura 8. Módulo Graph (primera parte).	58
Figura 9. Módulo Graph (continuación).	58
Figura 10. Módulo Circle.	61
Figura 11. Módulo Vis.	62
Figura 12. Entradas y salidas del sistema.	86
Figura 13. Ejemplo: la palabra seleccionada como primitiva es "suministrar".	87
Figura 14. Ejemplo: la palabra seleccionada como primitiva es "abastecer".	88

LISTA DE GRÁFICAS

Gráfica 1. Entradas totales contra entadas significativas	93
Gráfica 2. Resultados obtenidos por grafo y por método.	93

LISTA DE TABLAS

Tabla 1. Primitivas semánticas de Wierzbicka.	46
Tabla 2. Número de vértices en los grafos.	92
Tabla 3. Número de definidores, con diferentes algoritmos.	92
Tabla 4. Longitudes de los ciclos más cortos (ejemplo).	96
Tabla 5. Algunos ejemplos de datos de palabras relacionadas y sus frecuencias...97	
Tabla 6. Algunas primitivas halladas por el método 4.	97

Resumen

Esta tesis se sitúa en el campo de la lingüística computacional, en donde una de sus aproximaciones es la obtención de diccionarios computacionales a partir de los diccionarios tradicionales. Los diccionarios computacionales tienen una amplia aplicación en la resolución de problemas tales como la desambiguación del sentido de las palabras y la traducción automática. Un diccionario semántico computacional no puede contener ciclos en el sistema de definiciones, mientras que los diccionarios tradicionales existentes los contienen. Se presenta un algoritmo para la detección de ciclos y la selección de un conjunto mínimo de palabras a través de las cuales puedan definirse todas las demás palabras. Se describe una herramienta que ayuda al lexicógrafo a elegir tales palabras y corregir los defectos en el diccionario relacionados con los ciclos en las definiciones. Para tal efecto se realizó la construcción de una red semántica a partir de un diccionario tradicional, la depuración inicial de dicha red semántica, el diseño del algoritmo que encuentre un conjunto P mínimo de palabras y la aplicación de diversos métodos para obtener distintos conjuntos mínimos, justificando su aplicación y encontrando el conjunto más pequeño posible, dados los métodos empleados. Complementando este desarrollo se exponen las aplicaciones de los diccionarios computacionales, las diversas teorías semánticas y su relación con una forma de ambigüedad léxica llamada polisemia.

Abstract

This thesis is situated in the field of computational linguistic or automatic natural language processing. In this technology, dictionaries of a specific language, e.g., Spanish, have crucial importance. One of the possibilities to construct semantic dictionaries is conversion of existing large traditional explanatory dictionaries. However, a semantic dictionary for logical reasoning should not have vicious circles in the system of its definitions, while the traditional explanatory dictionaries do present this problem very frequently. In this thesis, we present an algorithm for detecting such circles and selecting a minimal (though not the smallest possible) set of words through which the definition of the rest of the words can be done without causing the vicious circles. Since for practical applications the number of elements in such a set should be as small as possible, we present also the some techniques to find such a set as small as we can (though still not guaranteed to be the smallest one). Basing on this algorithm we have developed a tool to help lexicographers to choose such words and correct the defects in the dictionary related to the circles in the system of definitions. The thesis gives an introduction to general and computational semantics to the extent as the task described above is concerned, and uses the methods of graph theory and computer science.

1 Introducción

Uno de los objetivos centrales de la lingüística computacional es el de permitir el uso de la lengua materna como medio de comunicación entre las computadoras y los individuos. Se trata de permitir que las personas puedan acceder a todas las facilidades ofrecidas por las computadoras, mediante órdenes expresadas espontáneamente con el vocabulario y la sintaxis de su propia lengua y, al mismo tiempo, que las computadoras presenten los resultados de sus aplicaciones en ese mismo idioma de manera natural y comprensible para las personas.

Dentro de las tareas en lingüística computacional ocupa un lugar preponderante la integración de diccionarios computacionales. Construir vocabularios definidores compuestos por primitivas semánticas, gracias a ciertos principios, es de gran importancia para contar con diccionarios computacionales que puedan aplicarse a aquellas tareas donde la semántica tiene un papel fundamental. La construcción de un diccionario computacional se puede hacer a partir de un diccionario tradicional, sin embargo ese tipo de diccionario presenta el problema de los ciclos, que se dan cuando una palabra tiene referencias a otras palabras y éstas a su vez a más palabras, dentro de las cuales puede llegarse de nuevo a la palabra de inicio. Por esta razón, es importante implementar algoritmos que, partiendo de un diccionario tradicional (que puede conseguirse con facilidad), puedan procesarlo para obtener un diccionario más adecuado para su uso computacional.

Este trabajo tiene como objetivo presentar algunos métodos y los algoritmos correspondientes para la detección de ciclos y selección de un conjunto definidor mínimo de palabras con la función de primitivas semánticas. Como marco teórico presento los métodos de procesamiento semántico, algunas de las corrientes más importantes en relación con las teorías semánticas (como la teoría de las primitivas semánticas); posteriormente presento el modelo matemático de un diccionario (representado como grafo), una descripción del algoritmo principal para la selección automática de las primitivas semánticas, un algoritmo de depuración de la red semántica, la implementación del algoritmo, la descripción de los experimentos realizados y la descripción e interpretación de los resultados.

1.1 Objetivo principal

Procesar un diccionario tradicional en español para detectar ciclos en las definiciones y, a través de distintos métodos, seleccionar conjuntos mínimos de palabras que puedan integrar un vocabulario definidor.

1.2 Metas particulares

- La construcción de una red semántica a partir de un diccionario tradicional.
- La depuración inicial de dicha red semántica.
- El diseño del algoritmo que encuentre un conjunto P mínimo de palabras.
- La aplicación de diversos métodos para obtener varios conjuntos mínimos, justificar su aplicación y encontrar el conjunto más pequeño posible, dados los métodos usados.

1.3 Diccionarios computacionales

Una de las tareas de la lingüística computacional es la de construir diccionarios para su uso por computadora. La utilidad más importante de los diccionarios computacionales se haya en su aplicación en la recuperación de información y en la traducción automática. En el primer caso, por ejemplo, un diccionario computacional puede utilizarse en la mejor resolución de consultas, analizando el contenido de los textos a través del diccionario. En la traducción automática, se aplicaría un diccionario de este tipo con el fin de lograr traducciones más correctas identificando el sentido preciso que tiene una palabra polisémica¹ en un idioma y su traducción a otro. Un método natural para la construcción de un diccionario semántico utilizable por los sistemas computacionales es la definición de unas palabras a través de otras, como se hace en los diccionarios explicativos tradicionales. Esta sustitución se repite iterativamente hasta llegar a conceptos cada vez más "simples"; en algún momento es necesario detener este proceso y así se llega a un conjunto de palabras "primitivas".

El concepto de primitivas semánticas ha despertado mucho interés entre los lingüistas, pues se ha discutido si existe realmente un conjunto de palabras a través de las cuales se puedan definir las demás palabras, y si estas primitivas semánticas forman un conocimiento humano innato independiente del idioma del que se trate.

Más allá de esta discusión, una forma de obtener diccionarios computacionales es a través de la construcción de vocabularios definidores compuestos por primitivas semánticas. Como se mencionó anteriormente, la construcción de un diccionario computacional a partir de un diccionario tradicional, es una aproximación que presenta el problema de los ciclos en las definiciones al sustituir iterativamente una palabra por su definición.

¹ Polisemia: Fenómeno muy común en el lenguaje en el que una misma palabra tiene múltiples significados. Para la teoría semántica y las aplicaciones semánticas es un problema muy importante, especialmente en la traducción y en la lexicografía.

La construcción de un diccionario computacional, en el que las definiciones de las palabras se realicen a través de otras palabras, despierta algunas dudas: ¿Se puede transformar un diccionario tradicional en un diccionario “para la máquina”?, ¿Qué problemas se enfrentarán?. En geometría, por ejemplo, se puede desarrollar cualquier término (digamos, *bisectriz*) sustituyendo en su definición cada palabra (digamos, *ángulo*) con la definición de ésta última. El sistema de definiciones se puede construir de tal manera que al repetir este proceso iterativamente, se llegará a una definición larga, compuesta sólo por los términos más "simples" como *punto y línea*.

Ahora bien, ¿qué va a pasar si en cada definición de un diccionario explicativo, sustituimos iterativamente cada palabra con su definición? ¿Cuáles palabras se deben considerar primitivas?.

El problema de las primitivas semánticas es uno de los problemas más interesantes en semántica. ¿Existe un pequeño número de palabras con las cuales se pueden definir todas las demás palabras? ¿Y si tales palabras existen, qué realidad se refleja en este fenómeno? ¿Existe la "*lingua mentalis*" (Wierzbicka, 1980) [45] en realidad o es un artificio creado en el marco de una teoría?.

Existen diferentes puntos de vista sobre este problema. La teoría de Wierzbicka (1980[45], 1996[47]) proclama un número muy reducido (alrededor de 60) de las llamadas primitivas, es decir, las palabras que se pueden usar para definir, en su totalidad, los cientos de miles de palabras que existen en el lenguaje.

Por otro lado, por ejemplo, Apresjan (1974[2], 1995[3]) sugiere usar un conjunto restringido de las palabras del lenguaje como la base para construir las definiciones de otras palabras, pero en su opinión este conjunto no sería muy pequeño.

Esta idea también se usa en la lexicografía práctica. En el diccionario *Longman dictionary of contemporary English* se usan en las definiciones sólo las

palabras del vocabulario definidor (*Defining Vocabulary o DF* en inglés), el cual es un vocabulario restringido. El tamaño de este vocabulario, conocido como *Longman defining vocabulary*, es alrededor de dos mil.

Sin involucrarse en la discusión sobre la naturaleza de las primitivas y el vocabulario definidor, notemos que un problema importante que impide considerar un diccionario tradicional como un diccionario “para la máquina” es la presencia de ciclos en las definiciones, esto es, cuando una palabra se define a través de otra y ésta a través de la primera. Por ejemplo:

abeja: insecto que segrega miel.
miel: sustancia que producen las abejas.

Pueden existir ciclos más largos, por ejemplo:

convenio: pacto, acuerdo.
acuerdo: pacto, tratado.
tratado: convenio.

Estas son definiciones tomadas (y un tanto simplificadas) del Diccionario Anaya de la lengua española (1997)[11]. Por supuesto, este sistema de definiciones no puede ser utilizado como un diccionario computacional.

Los ciclos en un diccionario explicativo representan un gran problema para cualquier trabajo semántico que involucra algún tipo de razonamiento. Un ejemplo es la resolución de la ambigüedad² en el sentido de las palabras. Este caso se presenta cuando una palabra tiene varios sentidos (polisemia y homonimia, como se explicará en la sección 2.2). Un método para encontrar el sentido preciso de una

² La desambiguación de las palabras es indispensable en la traducción automática y también en la recuperación de información, sobre todo en la resolución de consultas donde es necesario traducir las consultas a ciertos términos que un sistema pueda manipular para encontrar la respuesta más apropiada

palabra podría hacer uso de un diccionario con el fin de convertir una expresión compleja (como lo es cualquier expresión lingüística) en una expresión formada por términos "primitivos" o simples sobre las cuales se posea un dominio semántico que permita al sistema "entender" la pregunta y buscar una solución aceptable. Una forma muy simple que se nos podría ocurrir sería buscar las definiciones de los sentidos que esa palabra tenga en el diccionario, y, para elegir el sentido más indicado en función al contexto que que nos da el resto de la pregunta formulada, se podría realizar una búsqueda "a profundidad" en cada sentido, es decir, buscar ocurrencias de las palabras que conforman la pregunta en la definición en cada uno de los sentidos, aquel sentido en donde se presenten más ocurrencias, se considerará el sentido correcto. Por supuesto este método en muchos casos podría arrojar resultados insatisfactorios o no dar ninguna solución, pero hay otros métodos más complejos que se basan en la co-ocurrencia de términos recientemente explicada, aplicando tesauro, palabras clave, etc. Mientras más ciclos se encuentran en las definiciones menos oportunidad habrá de diferenciar unos sentidos de otros, además de que, computacionalmente, se vuelve muy costoso el manejo de los ciclos.

Concluyendo, un "buen" diccionario explicativo no debe tener ciclos. Para lograrlo, o bien hay que cambiar las definiciones, o bien, excluir las definiciones de algunas palabras del diccionario convirtiéndolas en las palabras primitivas (que posteriormente se puedan definir en un sistema computacional con otros medios, por ejemplo a través de un lenguaje de programación y no a través de otras palabras). En esta tesis se explora esta última posibilidad.

2 Métodos de procesamiento automático de información semántica

2.1 Teorías semánticas modernas

Existen distintas teorías semánticas modernas. Algunas de ellas están relacionadas con la aproximación clásica, otras más se inscriben dentro de la aproximación prototípica o bien de la aproximación relacional. Dado que la existencia de significados distintos para una misma palabra es un fenómeno que complica la descripción semántica y el uso de las palabras dentro de un contexto, la polisemia es un factor que es considerado y manejado por la mayoría de las teorías semánticas. En este inciso veremos algunas teorías semánticas modernas y su relación con la polisemia.

"La semántica relaciona al mundo extralingüístico con expresiones lingüísticas que lo describen"³. Una expresión léxica se aplica a un tipo extralingüístico a través de un proceso de abstracción; dicha abstracción es capturada en las teorías semánticas por medio de dos principios: la generalización, la cual reduce la polisemia para incrementar el poder explicativo de la teoría o bien

³ Ravin, Y. and Leacock, C. (2000) *Polysemy: An Overview*. [38]

el incrementar la polisemia, haciendo distinciones, para reunir el mayor número de detalles semánticos posibles.

2.2 Teoría clásica del significado

Para los clásicos o "Aristotelianos" una entidad individual es un miembro de una categoría conceptual. Cada categoría se define por 1) condiciones necesarias o características definitorias y 2) condiciones suficientes o propiedades nucleares.

Las categorías conceptuales pueden ser vistas como un conjunto de listas de características conectadas por operadores lógicos. Estas categorías se organizan jerárquicamente, donde conceptos del mismo nivel de la jerarquía heredan y comparten las propiedades esenciales o nucleares de los conceptos que están por arriba de ellos, pero poseen un conjunto de características que son mutuamente exclusivas. Un objeto pertenece o no pertenece a una categoría conceptual; el problema que se presenta con los objetos que son ambiguos se soluciona eligiendo "arbitrariamente" alguna de las categorías posibles.

Esta forma de entender la semántica es retomada, hoy en día, por numerosos investigadores, gracias a su enfoque jerárquico y clasificadorio, y por el hecho de que se fundamenta en la lógica. Una corriente moderna que puede inscribirse dentro de la aproximación clásica es la de Anna Wierzbicka (*Natural Semantic Metalanguage - NSM*) (Wierzbicka 1996) [47], quién indica que el problema del análisis semántico no se encuentra en hacerlo en función de características necesarias y suficientes, sino en que implícitamente se crea que existe una correspondencia observable y mensurable de éstas con aspectos certeros de la realidad externa

Tradicionalmente se ha visto a la polisemia como diversidad de significados para una misma palabra, a la sinonimia como un mismo significado

para varias palabras. Por otro lado, la homonimia se refiere a varios significados para una misma palabra, la cual sin embargo proviene etimológicamente de orígenes distintos y por lo tanto los sentidos y los dominios en el que se emplean la palabra también son muy distintos. Como ejemplo de ello tenemos en español la palabra "estela", con dos sentidos diversos, uno de los cuales hace referencia al rastro que deja un objeto al atravesar un fluido, y el otro, a un monumento monolítico de carácter conmemorativo. El primer sentido deriva del latín *aestuaria*, mientras que el segundo proviene del griego *stèle*. Un ejemplo de polisemia es la palabra "brillante" (del verbo brillar, éste último del italiano *brillare*), que puede tener varios sentidos, como adjetivo significa que brilla, o que es admirable o sobresaliente. Tiene otro sentido más, que es, por supuesto, el de un diamante tallado. Todos estos sentidos tienen cierto parecido semántico. Se podría decir, según la aproximación de Katz (1972) [24], que comparten marcadores, esto se debe a que provienen etimológicamente de la misma palabra (el concepto de marcadores se explicará más adelante). Cada teoría del significado caracteriza y maneja a la polisemia en sus propios términos.

Homonimia			Polisemia		
estela	I. Del lat. <i>aestuaria</i> .	1. Rastro que deja un objeto al atravesar un fluido.	brillante	I. Del ital. <i>brillare</i> .	1. Que brilla
	II. Del gr. <i>stèle</i> .	1. Monumento monolítico de carácter conmemorativo.		2. Admirable o sobresaliente.	
					3. Diamante tallado por sus dos caras

Figura 1. El uso de notaciones lexicográficas para distinguir diferentes sentidos de una palabra.

Como se mencionó en el párrafo anterior, una corriente moderna de la teoría clásica está en la teoría intencionalista y racionalista de la semántica, desarrollada por Katz y Fodor (1963) [25] y más adelante por Katz (1972)[24]. Ellos parten de la teoría estándar de Chomsky y su objetivo es articular los principios de una teoría semántica universal que explicaría el sistema de reglas internalizadas que constituyen la habilidad o capacidad lingüística del hablante nativo. Semánticamente, esto se refiere a las reglas que relacionan las estructuras lingüísticas del hablante nativo.

Katz (1972) [24] comienza el debate haciendo a un lado la discusión ontológica sobre el significado. Para él, la meta de una teoría semántica es hallar un sistema general para representar el significado. Este esquema provee un mecanismo para definir cómo la forma lógica o significado de proposiciones u oraciones está definido por los sentidos de sus componentes, y de las relaciones sintácticas entre esos componentes.

Una teoría semántica provee una representación semántica para cada sentido de cada elemento léxico o palabra. Las palabras ambiguas se representan en un diccionario con más de un sentido o representación de sentidos.

Las representaciones semánticas de sentidos tienen unidades más pequeñas llamadas marcadores semánticos, que es algo más o menos equivalente a las propiedades nucleares y definitorias, y reflejan los conceptos componentes que constituyen un sentido particular.

Los marcadores semánticos no son necesariamente primitivos, esto es, pueden subdividirse en otros marcadores semánticos.

Los marcadores semánticos permiten inferencia, pues al heredarse la serie de características que constituyen una clase particular, se puede inferir que un instancia de una clase en especial también es un miembro de la clase situada en un nivel superior, pues todos sus marcadores semánticos se encuentran también en

dicha clase. Ejemplo: En la oración "Una silla está en la habitación" se deriva que "Un objeto está en la habitación" como una inferencia válida

Sobre este concepto también es conveniente señalar lo siguiente:

- Los sentidos relacionados, se refieran o no de la misma palabra, comparten marcadores semánticos (uno o más).
- Existe una medida llamada similitud semántica que está en función del grado en que se comparten marcadores semánticos.
- La sinonimia es aquella en la que la compartición de marcadores semánticos es completa.
- La ambigüedad se da cuando una palabra tiene más de una representación semántica.

Katz no distingue polisemia de homonimia, su objetivo no es estudiar el origen de los diferentes sentidos, tampoco explica lo que es la polisemia regular, fenómeno a través de cual un elemento léxico con una representación de sentido adquiere otra representación que difiere de la primera de maneras predecibles.

Apresjan define la polisemia como la similitud en las representaciones de los sentidos de una palabra, basta que cada significado esté ligado por lo menos a otro, no es necesario que en todos exista una parte común. Apresjan estudió el caso en que las palabras son sistemáticamente ambiguas, como el caso, en inglés de "la construcción está completa", pues esta oración puede significar o bien que el objeto físico construido está terminado o bien que la jornada de trabajo en el espacio físico donde se realiza esta acción ha concluido. En español podríamos dar el ejemplo de la oración "En la reunión tocaron música" también con dos sentidos: que unas personas realizaron la acción de tocar instrumentos o bien que a través de un aparato reproductor, se escuchó música en dicha reunión.

La polisemia regular está gobernada por procesos que son productivos, muy similares a los de formación de palabras, que se desarrollan conforme a reglas, y que son procesos predecibles. Ejemplos de estos procesos son: la metonimia (responsable de que surjan nuevos sentidos como sucedió con la palabra "pie" en "al pie de la montaña") o la relación entre el nombre de un contenedor y la cantidad que se le adjudica a dicho contenedor, como en "Una taza de azúcar".

Aunque el fenómeno de polisemia regular no es un problema mayor dentro de la aproximación clásica, la distinción entre sentidos sí lo es. La teoría clásica postula que con cada diferencia conceptual se tiene un nuevo sentido, pero éste implica el peligro de la proliferación infinita de sentidos. Katz acota este peligro indicando que el significado debe quitar las características variables de las cosas y enfocarse sólo en las características invariables por virtud de las cuales una cosa es *cierta* cosa y no otra. De otra manera ninguna palabra se podría volver a usar con el mismo significado.

Esta afirmación es importante porque permite la abstracción o indeterminación, aún dentro de la aproximación clásica, en la construcción de representaciones semánticas. Las otras teorías no clásicas llevan esta idea de indeterminación mucho más allá, haciéndose preguntas de este tipo: ¿porqué el implemento para comer no forma parte del significado de "comer", porqué no es esencial en muchos casos y en algunos sí?. Ravin investiga en este sentido, aplicándolo a los verbos; para él no hay un criterio claro para decidir que aspectos del mundo real son relevantes en la semántica particular de un verbo. Sin embargo, Ravin propone una metodología, la cuál se basa en la idea de que en las representaciones semánticas deben reunirse todas las propiedades semánticas y las relaciones de expresiones lingüísticas (por ejemplo *break into pieces* y *shatter*, con significados idénticos, deben tener representaciones idénticas, estableciendo restricciones en la representación de las palabras constituyentes *break, into, pieces*).

Anna Wierzbicka (*Natural Semantic Metalanguage, NSM*) [45] propone una metodología para distinguir una palabra polisémica de la que no lo es e identificar su significado. Consiste en asumir que hay un solo significado, construyendo una definición y después esa definición se compara con el uso de esa palabra en el mundo real; si es necesario (ya que esta palabra no funcione para algunos casos de uso común) se postula otro significado, y así sucesivamente.

Ella usa la idea intuitiva de que la representación semántica de una palabra puede ser sustituida por la palabra en todos sus usos (intercambiabilidad definición-palabra) El problema de esto es que muchas veces es difícil obtener las intuiciones semánticas de los hablantes de una lengua y que no hay pruebas objetivas confiables que ayuden en esta labor.

NSM se basa en la evidencia de que existe un conjunto pequeño de significados universales y básicos, llamados primitivas semánticas, que pueden encontrarse como palabras u otras expresiones lingüísticas en todos los lenguajes. Este núcleo puede ser usado como una herramienta para la lingüística y para el análisis cultural: explica palabras complejas, palabras específicas a una cultura y construcciones gramaticales, y articula valores específicos de una cultura y actitudes o *scripts culturales*⁴ en términos que son claros y traducibles. La teoría también provee las bases semánticas para una gramática universal y para la tipología lingüística. Tiene aplicación en la comunicación, lexicografía, enseñanza de un idioma y el estudio de adquisición de lenguaje en la infancia, semántica legal, etc.

⁴ Un *script* cultural es una forma de llamar a distintas convenciones de discurso "locales" usando el metalenguaje de primitivas semánticas inventado por Wierzbicka.

2.3 Aproximación prototípica

En un principio, Wittgenstein (1958) [49], refiriéndose a los sentidos de las palabras, señaló que cualquier categoría tiene fronteras difusas y los significados exhiben reminiscencias familiares con elementos comunes que se sobreponen y cruzan de la misma forma que los rasgos familiares aparecen en los miembros de cada familia de personas. La aproximación prototípica está más cercana a la psicología que a la lógica y a la filosofía (como la aproximación clásica). Rosch, en la década de 1960, demostró que las personas no categorizan objetos en la base de condiciones necesarias y suficientes sino en la base de la reminiscencia de los objetos a un miembro prototípico de la categoría.

Esta corriente respeta la jerarquía entre conceptos pero ve la membresía a una categoría como un asunto de grados, algo relativo, que se mide respecto a un "prototipo" o tipo ideal.

Según Fillmore (1982) [15], un sentido está formado por la disyunción de otros sentidos; cuando aparecen todos los sentidos en una oración, el sentido del referente es más prototípico, si no se cumple ninguna condición, el tratamiento es anómalo. Ejemplo:

La palabra "*escalar*" está compuesta por la disyunción: "*subir*" ó "*tregar*". En la oración "*Juan escaló la montaña*" el uso de "*escalar*" es prototípico (se cumplen ambas condiciones, es decir, tanto "*subir*" como "*tregar*" son aplicables). En "*Juan escaló hacia abajo la montaña*" el sentido "*subir*" no es aplicable, por lo que el uso no es totalmente prototípico. En la frase "*La serpiente escaló el árbol*" el sentido "*tregar*" no es aplicable, por lo tanto, el uso no es prototípico.

2.4 Aproximación relacional

Según esta aproximación las palabras están organizadas acorde a sus significados usando relaciones semánticas o enlaces para formar redes semánticas. Hace explícita la organización estructural que está implícita en otros modelos y describe cómo elementos de un dominio están relacionados entre sí. La aproximación relacional trata de ubicar cada palabra del lexicón o diccionario, en un lugar particular de esta red, dependiendo del dominio y conforme a cierto tipo de relaciones, sobre todo la relación de sinonimia, antonimia, hyponimia (relación jerárquica - 'is_a') y troponimia (se da entre verbos, con variedades de un mismo verbo, como la relación entre "trotar" y "correr"). Un autor importante de esta corriente es Fellbaum (1990)[14], (1998)[16].

2.4.1 Polisemia en la aproximación relacional

La representación de la polisemia en esta corriente es problemática, pues sentidos con polisemia regular (ya mencionada en el apartado 3.2), pueden quedar alejados en la red semántica conceptual. Mel'cuk (1988)[32] usa índices para establecer relaciones de polisemia regular o no-regular usando una relación especial de similitud.

2.5 Problemas relacionados con polisemia

2.5.1 Polisemia desde el punto de vista computacional

La polisemia es uno de los principales problemas del procesamiento de lenguaje natural; su estudio ha estado muy ligado al de la traducción automática. El tratamiento computacional de la polisemia busca relacionar expresiones a sus significados correspondientes automáticamente. Una de las formas en que esto se

resuelve es contar con diccionarios lémbles por computadora, de conocimiento lexicográfico de los diferentes sentidos de las palabras polisémicas y otra forma es simular el entendimiento humano usando procedimientos estadísticos para capturar patrones de co-ocurrencias de las palabras en el contexto. Bar-Hillel (1964)[4], quien en un principio estuvo a favor de la traducción automática concluyó que no era posible identificar el sentido correcto automáticamente pues la desambiguación requiere de una caracterización completa del conocimiento del mundo, lo cual es inabarcable.

Katz y Fodor (1963)[25] publicaron su teoría de interpretación semántica, la cual establece unas estructuras que están contenidas en el lexicón mental, independientes del conocimiento del mundo. Para esta teoría crearon algoritmos y métodos para la desambiguación de sentido automático, logrando desambiguar 671 palabras en un período de 7 años, una cantidad de trabajo demasiado costosa. No obstante lo anterior hay un interés renovado, gracias a los diccionarios computacionales y al análisis de enormes corpus de texto con la capacidad computacional actual.

El grado óptimo de abstracción sobre múltiples significados está determinado frecuentemente por la meta computacional. En la traducción automática, consideramos que hay distintos sentidos si existe una elección léxica en el lenguaje destino, por ejemplo, la palabra "*know*" en inglés en la oración "*know a person*", se traduce en al francés "*connaitre*", mientras que "*know a fact*" se traduce en el mismo idioma como "*savoir*". El grado de abstracción está determinado en ocasiones por los límites de la propia tecnología. En la recuperación de información a gran escala, por ejemplo, la polisemia es sólo un aspecto de un problema mayor de ambigüedad, el cual incluye ambigüedad semántica (el verbo "*table*" contra el sustantivo "*table*") y la homonimia ("*river bank*" contra "*bank*" como institución financiera).

En los 80's el uso de diccionarios leíbles computacionalmente (MRDs-*Machine Readable Dictionaries*) se propuso para la desambiguación de sentidos. Lesk (1986) [30], enlaza definiciones en diccionarios si éstos comparten palabras, los algoritmos de Lesk escogen el sentido contenido intersectando palabras, por ejemplo, si "*pine*" tiene 2 sentidos y "*cone*" tiene otros tres, tenemos 6 sentidos posibles resultantes para las oraciones que tengan esas dos palabras, se elige el sentido en el que aparezcan mayor número de palabras en común.

Otra forma de realizar la desambiguación de sentidos es el uso de códigos temáticos. El código temático de un texto es aquél que está asociado con el mayor número de palabras del mismo texto. El sentido elegido es aquel asociado con el código temático. Otra forma es la desarrollada por Guthrie et al. (1991) [21] en la que se obtienen "vecindarios" para las palabras polisémicas, a través de la búsqueda de co-ocurrencias en las definiciones y oraciones de ejemplo de todas las palabras que comparten el mismo código temático.

Recientemente los métodos de desambiguación basados en el corpus están en boga. Actualmente se puede usar corpora de millones de palabras, anteriormente el análisis a gran escala de palabras que co-ocurren con las palabras polisémicas no estaba al alcance. La identificación de sentidos basándose en el corpus, esto es, el análisis estadístico basado en el corpus, de co-ocurrencia de patrones, se convirtió en la corriente principal de la lingüística computacional. El entrenamiento de los sistemas de identificación se hace a través de oraciones de ejemplo para cada sentido de una palabra (desde 50 hasta cientos de ellas) y con ellas el sistema crea un modelo para cada sentido. Una vez que el modelo ha sido creado, el sistema escoge el modelo más apropiado para un nuevo uso de la palabra. Esto lo realiza a través de una medida de similitud entre las características del modelo y aquellas del contexto de la nueva ocurrencia. Miller y Leacock caracterizaron los tipos de información contextual que una computadora puede extraer del corpus. Así, llamaron contexto local a las que están cerca o adyacentes a la palabra polisémica mientras que el contexto tópico, son sustantivos

que co-ocurren con un sentido en la discusión de un tópico dado. Los sistemas de identificación automática de sentido que hace explícito el uso de contextos tópicos o locales los desarrollaron Towell y Voorhees(1998)[43] y Leacock et al. (1998)[29].

En la reunión de Senseval⁵ en 1998 se llevó a cabo la desambiguación de 335 palabras polisémicas, los resultados condujeron a lo que Bar-Hillel llamaron la "falacia del 80%", a partir del cual el esfuerzo para desambiguar el restante 20% es mayor que el realizado con antelación.

Otro aspecto problemático de esta aproximación es el llamado "cuello de botella de adquisición del conocimiento"(Gale et. al. 1992)[18], en el que el material para entrenar un programa requiere, para cada palabra polisémica, de un corpus del contexto que tiene que ser dividido en diferentes sentidos, y el esfuerzo manual requerido debe hacerse en cada caso sin posibilidad de aprovechar el esfuerzo realizado en otras palabras.

Shütze toma una dirección distinta para hacer una identificación de sentido basado en el corpus, donde su preocupación central es diseñar un algoritmo que sea psicológicamente plausible y generalmente aplicable. En lugar de asignar una palabra polisémica a un sentido discreto, Schutze reúne sentidos de palabras que son similares, pues comparten contextos similares, y después define estos *clusters* como sentidos de palabras.

Otros autores como Dolan, Vanderwende y Richardson [11] combinan las dos aproximaciones principales: el uso de MRDs y las basadas en el corpus. Como Shutze, utilizan una medida de similitud en el contexto de las palabras para

⁵ Senseval es una conferencia para evaluar el estado del arte en relación con los sistemas de desambiguación de sentidos, basados en el corpus (*International Workshop on Evaluating Word Sense Disambiguation Systems*). El primer SENSEVAL se realizó el verano de 1998, culminando en un taller realizado en Herstmonceux Castle, en Sussex, Inglaterra, del 2 al 4 de septiembre de dicho año.

determinar si están cercanas en significado. Sin embargo, los contextos que ellos usan son muy especializados y, además, son enlaces en una red semántica llamada MidNet, la que se obtiene del análisis de las definiciones de un MRD. MidNet no trata de clasificar sentidos individuales de palabras, sino reconocer conceptos que son similares a aquellos a los que se les da un uso nuevo. La aproximación de Schütze y Dolan puede ser útil para la recuperación de información donde la tarea es vincular el contexto de la consulta a contextos similares en la base de datos de los documentos, pero no es obvio cómo esas aproximaciones pueden ser usadas en aplicaciones como la traducción automática, sin una codificación extensa y manual.

En la lingüística computacional es más fácil manejar los homónimos que las palabras polisémicas, pues el contexto de los homónimos tiene un vocabulario muy distinto, mientras que cuando hay polisemia, puede ser muy similar. Por otro lado, la recuperación de información, donde a un usuario se le dan a escoger varios documentos en función a una consulta, es un ambiente mucho menos restrictivo que la traducción automática.

El campo de la lingüística computacional crece en dos direcciones. La colección de textos en línea está creciendo en tamaño y así la demanda de procesamiento a gran escala. La investigación a gran escala se especifica como una meta importante de muchos proyectos de investigación. La otra dirección es la aplicación de computadoras a nuevos dominios, los editores y compañías de computación experimentan con diccionarios en línea, bibliotecas digitales, métodos para navegar, buscar y acceder a datos relevantes en ambientes multilingües, etc.

2.6 Procesamiento automático de los diccionarios explicativos

La medición de la relación semántica tiene como fin revelar que tan parecidas son dos palabras entre sí y en qué sentido son parecidas. Esto tiene una aplicabilidad muy amplia en el procesamiento del lenguaje natural, por ejemplo en la desambiguación de términos polisémicos. Se puede encontrar el sentido correcto de una palabra polisémica al calcular la relación semántica de la palabra en cada uno de sus sentidos, a una tabla de palabras no ambiguas que la rodean en el texto, y escogiendo el sentido que presenta el valor acumulativo de relación más alto. Otra aplicación es determinar la estructura de un texto, pues el conocimiento de la relación entre palabras permite identificar secuencias o cadenas de palabras altamente relacionadas que a su vez pueden hacerse "corresponder" (mapearse) en el texto y así encontrar los límites entre segmentos de texto que forman dominios o unidades "tópicas". Si un algoritmo es capaz de formar estas cadenas y diferenciarlas en función a su "fuerza", podrían hacerse resúmenes de un texto dado extrayendo segmentos de texto que correspondan a las cadenas más fuertes que cierto "umbral" o valor mínimo. Un uso evidente dentro del área de recuperación de información es reemplazar la equivalencia léxica convencionalmente usada, pues en lugar de recuperar documentos gracias únicamente a la ocurrencia de los términos de la consulta en el documento, podemos incluir documentos que contienen términos que están semánticamente relacionados a los términos de la consulta. En el reconocimiento de texto y de los mensajes del habla, cuando aparece un lexema irreconocible, la interpretación del mismo puede realizarse escogiendo el candidato más cercanamente relacionado a un subconjunto de lexemas reconocidos con anterioridad.

Entre los pioneros de la ola contemporánea de investigación, están Osgood (1952)[35], Quillian (1968)[37], y Collin y Loftus (1975)[9]. Osgood definió un "diferencial semántico" como un intento de representar palabras como entidades en un espacio n-dimensional, donde la medición de la distancia entre ellas proviene de nuestro conocimiento de la geometría euclídeana. Osgood encontró

que este sistema se sostenía en las "emociones connotativas" que se asocian a una palabra más que a su "significado denotativo" y lo discontinuó. Quillian, Collins y Loftus iniciaron entonces una aproximación más procedural llamada "activación extensiva" o "*spreading activation*", la cual continúa en boga por algunos investigadores.

Hace dos décadas McGill y colegas determinaron 67 medidas de similitud para la recuperación de información en general, la medición de la relación semántica es el principal motor de dicha investigación.

Por lo menos hay tres términos para designar a esta característica: relación, similitud y distancia. Resnik (1995)[39] ve la similitud como un caso especial de relación. La relación semántica o relacionismo semántico agrupa varios tipos de relaciones: meronimia (ventana-casa), antonimia (caliente-frío), y asociación funcional (océano - barco). El término distancia semántica puede ser más confuso, pues puede aludir tanto a la similaridad como a la relación, es mejor verlo como lo inverso al relacionismo y no solamente lo inverso a la similitud.

2.6.1 Aproximaciones basadas en diccionarios.

Los diccionarios representan la fuente de conocimiento lingüístico más común. Por esta razón es natural que se haya pensado en aplicar algún diccionario al problema de determinar la distancia semántica entre las palabras. El primer diccionario que estuvo disponible en un formato leíble por computadora y el que probablemente siga siendo el más usado actualmente (Guthrie et. al. 1996)[22] es el *Longman Dictionary of Contemporary English* (LDOCE). Una característica especial es que posee un vocabulario definidor, el *Longman Defining Vocabulary* (LDV) de 2851 palabras, que se emplean para formar las 56,000 definiciones de palabras restantes. Este vocabulario definidor se basó en la investigación de West (1953)[44].

Kozima y Furugori (1993)[27] estructuraron una red semántica a partir de LDOCE en la que a cada nodo corresponde una palabra del diccionario. Los enlaces van de cada nodo *n* a otros nodos, que son las palabras contenidas en la definición de *n*.

Por ejemplo, la palabra abanicar aparece en el diccionario de la siguiente manera:

Abanicar

I. Hacer aire con un abanico

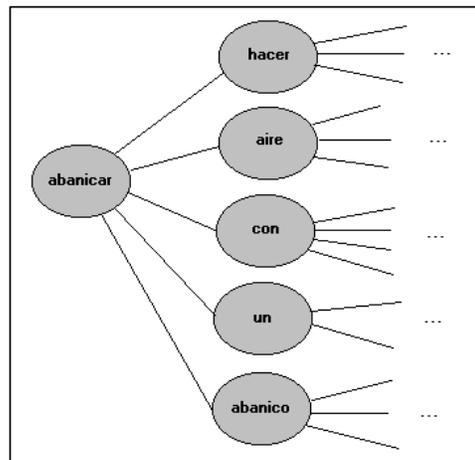


Figura 2: Parte de una red semántica.

El nodo que representa a la palabra “abanicar” tiene enlaces con los 5 nodos que representan a las palabras que forman parte de su definición. A su vez, estas palabras deberán tener enlaces a los nodos de las palabras que componen sus definiciones respectivas. Si la palabra “abanicar” aparece en la definición de otra u otras palabras, existirán enlaces desde los nodos de éstas palabras hacia el nodo de “abanicar”.

Un buen diccionario debe ser un sistema cerrado de lenguaje natural, donde todas las palabras que se usen en la definición sean a su vez definibles por el mismo diccionario.

Si existe un vocabulario definidor, le corresponde la parte más densa de la red, mientras que las palabras que no son definidoras no están ligadas entre sí, por lo tanto se encuentran en la periferia.

La forma en que estructuraron esta red semántica fue, en un principio, separando las palabras que pertenecen al LDV. Al subdiccionario resultante lo llamaron "Glosema", y estuvo compuesto de 2,851 entradas y 101,861 palabras en total. Cada entrada del Glosema está compuesta del término principal o "*headword*", la clase de palabra, o "*word class*", y una o más unidades correspondientes a las definiciones de los sentidos numerados para esa palabra, en el LDOCE. Cada unidad consiste en una "*head-part*" que corresponde al "*genus*" y una o más "*det-parts*" que corresponden a la diferencia.

Posteriormente el Glosema se tradujo subsecuentemente en una red semántica llamada "Paradigma". El Paradigma cuenta con 2,851 nodos que corresponden a las entradas del Glosema, interconectados con 295,914 ligas no etiquetadas. Cada nodo en el paradigma incluye un "*headword*", un "*word-class*" y un "*activity-value*" y se encuentra ligado a las palabras que aparecen en la definición de la entrada del Glosema (aproximadamente 104 ligas por nodo). Las ligas pueden ser de dos tipos, la "*référants*" es la que refleja la intención del "*headword*" del nodo. Contiene varios subconjuntos llamados "*subréférénts*", cada uno de los cuales corresponde a una unidad del Glosema (sus enlaces conectan cada nodo a los nodos que representan las palabras de la unidad). Estas ligas pueden llamarse "ligas salientes" o "*outgoing links*". La "*référé*" de un nodo *n* nos da información sobre su extensión, al ligar *n* a los nodos que se refieren a él en la definición respectiva del Glosema; a estas ligas se les puede llamar "entrantes" o "*incoming*".

Hecho esto, la similitud entre palabras en el LDV se calcula a través de la función de activación "*spreading activation*". Cada nodo tiene cierta "actividad", expresada a través del campo "*activity-value*" o v_n , la cual es recibida y transmitida a través de las ligas de los nodos. Se calcula como sigue:

$$v_n(T+1) = \phi\left(\frac{R_n(T) + R'_n(T)}{2} + e_n(T)\right) \cdot \sigma \quad (1)$$

donde:

$R_n(T)$ es la actividad compuesta de los nodos referidos ("*référant*") por n en el tiempo T

$R'_n(T)$ es la actividad compuesta de los nodos referidos ("*référé*") hacia n en el tiempo T

$e_n(T)$ es la actividad impartida en el tiempo T , en " n " desde fuera

ϕ es una función que limita la salida al intervalo $[0,1]$

El algoritmo que calcula la similitud $\text{sim}_{\text{KF}}(w_k, w_l)$ entre las palabras w_k y w_l es:

1. Reiniciar valores de actividad de todos los nodos de la red.
2. Activar el nodo k correspondiente a la palabra w_k , con la fuerza $e_k = s(w_k)$ por 10 pasos para obtener el patrón de activación $P(w_k)$. La significancia de w_k , $s(w_k)$ se define como la información normalizada de la palabra w_k , en el corpus de 5,487,056 palabras.
3. Observar $a(P(w_k), w_l)$, el valor de actividad del nodo l en el patrón $P(w_k)$ (calculado según la fórmula (1)). La similitud es:

$$\text{sim}_{\text{KF}}(w_k, w_l) = s(w_l) \cdot a(P(w_k), w_l).$$

$$s(w_1) = \frac{-\log(n(w_1)/N)}{-\log(1/N)}$$

Donde:

$n(w_1)$ es el número de veces que aparece w_1 en el corpus.

N es el número de palabras del corpus.

Este algoritmo es para LDV x LDV $\rightarrow [0,1]$, que representa solo el 5% de LDOCE.

Para que abarque todo LDOCE se aplica la siguiente fórmula:

$$\text{sim}_{\text{KF}}(W, W^n) = \psi\left(\sum_{w \in W^n} s(w) \cdot \alpha(P(W), w)\right)$$

Donde:

$P(W)$ es el patrón resultante de la activación de cada $u_i \in W$ con la fuerza $s(u_i)^2 / \sum s(u_k)$ para 10 pasos o etapas y ψ es una función que restringe la salida al intervalo $[0,1]$.

Comentario [GRL1]:

El método de Kozima y Furugori, así como los de Osgood, Morris y Hirst son métodos libres de contexto o estáticos porque miden la distancia de las palabras sin tomar en consideración la influencia del contexto. Posteriormente Kozima e Ito derivaron, del método explicado arriba, otro que es más dinámico y sensible al contexto; la medida que utilizan toma en consideración la "dirección asociativa" de un par dado de palabras.

2.7 Recuperación de información con los métodos de lingüística computacional

Semántica léxica

Actualmente la semántica léxica juega un papel muy importante en la lingüística computacional, junto con los formalismos gramaticales para el análisis y la generación del lenguaje, así como la producción de una representación semántica del discurso y del enunciado.

El papel central de la semántica léxica en la lingüística computacional puede explicarse por el hecho de que cada entrada léxica contiene una parte considerable de información relacionada al sentido de la palabra que representa. El campo del procesamiento del lenguaje natural se ha apoyado en algunos conceptos de la psicolingüística, sobre todo en aquellos aspectos relacionados a la forma en que se organizan los conceptos en la memoria, y también a la forma en que esta información es recuperada del "lexicón mental". La investigación en psicolingüística tiene mucha influencia en relación con los modelos de representación del conocimiento para los sistemas de inteligencia artificial. Los seres humanos procesamos lexemas para nombrar objetos, elaborar pensamientos y reconocer palabras escritas, escuchadas o pronunciadas. Según Collins y Quillian (1972)[8], a través de su modelo de organización de la memoria, los conceptos forman parte de una red de nodos "tipo" y nodos de "ocurrencia", y las relaciones que los asocian (*and, or*), por ejemplo, según algún diccionario cualquiera, la definición de la palabra "*planta*" podría ser la siguiente:

Planta: Estructura viviente que no es un animal, que con frecuencia tiene hojas, toma su alimento del aire, del agua y de la tierra.

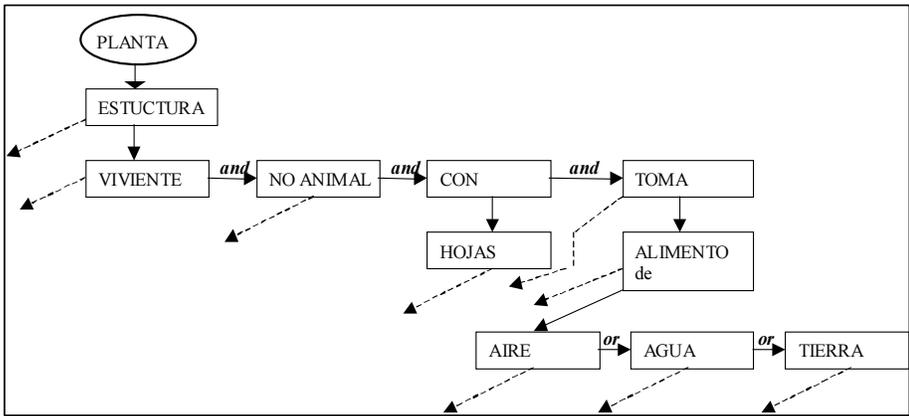


Figura 3: Representación del significado de la palabra "planta".⁶

Básicamente una red semántica es un grafo con nodos que representan conceptos, y aristas que son relaciones binarias entre los nodos. En este modelo los conceptos no son primitivos. El modelo de Collins y Quillian (1972)[8] está centrado en un almacenamiento de memoria a largo plazo, en otras palabras, en "conocimiento enciclopédico". Otros modelos distinguen entre memoria a largo plazo y memoria de trabajo o a corto plazo. Existen modelos que no reconocen que exista una diferencia entre ambas. Para Collins y Quillian, el modelo debe basarse en la economía de espacio, mientras que otros modelos (Conrad, 1972)[10] prefieren tener como principio la eficiencia, a través de la flexibilidad de acceso, aunque haya redundancia en la información.

El modelo de Miller y Johnson Laird (1976)[33] destaca el papel de la percepción. En su modelo la percepción es una actividad estructurada que destaca características distintivas de un objeto; a la representación de la información relevante asociada a un concepto la llaman "esquema". El nivel conceptual es

⁶ Los nodos tipo se representan con elipses, los nodos de ocurrencia con rectángulos, las relaciones asociativas con flechas o arcos entre los nodos. Cada nodo de ocurrencia se relaciona con un nodo tipo, en este diagrama estas relaciones aparecen con flechas punteadas.

donde convergen los esquemas perceptuales y los elementos léxicos. Un concepto léxico consiste en una etiqueta (por ejemplo, "tabla"), un esquema, y las reglas que gobiernan el comportamiento sintáctico de la etiqueta.

El esquema integra información funcional y perceptual y podría incluir información almacenada que no tiene ninguna consecuencia perceptual directa, como se ve en la ilustración 3:

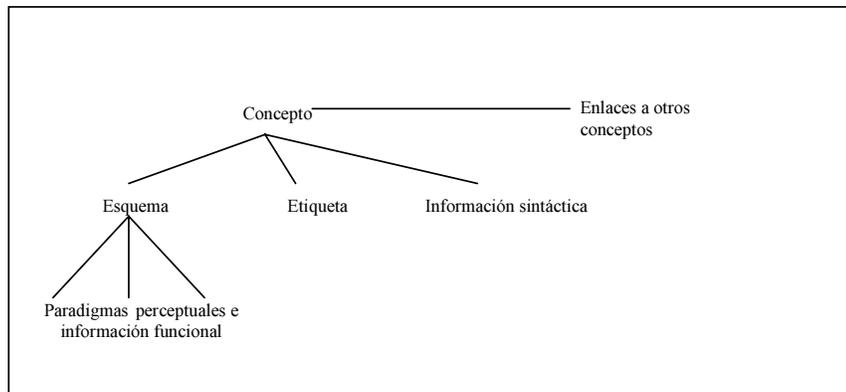


Figura 4: Modelo de Miller y Johnson Laird.

El significado lingüístico de los elementos léxicos se deriva de las relaciones que pueden darse entre ellos, comúnmente: IS-A (es un(a)), HAS-A(tiene un(a)), IS-IN(esta en).

Según estos autores, un lexicón mental tiene dos componentes:

1. Un componente léxico como su inclusión en una relación de clase todo-parte
2. Un componente conceptual, que se refiere a las características perceptuales, información sintáctica mínima, información "mnésica" (emoción, funcionalidad, etc.).

Según Allport (1985)[1], las investigaciones recientes se inclinan más a no establecer una separación entre la memoria a corto plazo y la memoria a largo plazo, y tampoco a considerar que la percepción es la entrada a una "caja negra" de la que se obtendrá la memoria/conocimiento. Contrariamente a esto, opinan que la memoria es una capacidad que interviene en cada nivel de actividad de la "caja negra" y que la percepción y la memoria/conocimiento están interrelacionadas. La memoria intervendría así en aspectos sensoriales, motores, y en aspectos cognitivos más elevados.

Actualmente se cree que el acceso a la memoria es distinto dependiendo de si se trata de reconocimiento o de producción.

El acceso léxico se refiere a cómo reconocemos que un elemento léxico es simple o compuesto y de qué manera obtenemos esta información del sitio donde está almacenada. Uno de los aspectos que se toman en cuenta es que no solo tenemos acceso a las palabras individuales, sino también a las relaciones entre ellas, en términos de sintaxis y del significado del enunciado.

Todos estos modelos tienen en común la búsqueda de un sistema para detectar palabras. En este sentido se parte de dos hipótesis, una en la que las palabras se recuperan siguiendo un procesamiento secuencial (como se hace a través de un diccionario), modelo de Forster (1976)[17], y otra en la que el procesamiento es paralelo y se realiza gracias a la activación de palabras dentro de un tratamiento interactivo del lenguaje (Morton, 1982 [34]; Marslen-Wilson, 1984 [31]).

Forster también da algunas claves de cómo estructurar lexicones a través de su "*semantic priming*", esto es, que el tiempo de reacción para el reconocimiento de estímulos es menor, cuando un elemento léxico ha sido presentado después de otro al que está relacionado, semántica y contextualmente, es decir, las personas reconocen más rápido la palabra "enfermera" después de "doctor", que la palabra "abogado".

Morton (1982)[34] ha desarrollado el modelo "logonen", en el que define a los *logonens* como unidades relacionadas a los elementos léxicos y que permanecen activados durante el proceso de recuperación. Morton sostiene que un sistema de reconocimiento de palabras, para ser operacional, necesita de toda la información del contexto y que la recuperación de los elementos debe hacerse en paralelo.

Marslen y Wilson (1984)[31] desarrollaron el modelo "cohort", un sistema en que la identificación de palabras no es resultado de la activación selectiva de *logonens*, sino de la eliminación progresiva de ellos. Para Marslen -Wilson la elaboración de significado inicia tan pronto como una oración comienza.

En términos de la producción, el acceso léxico depende de cómo asociamos un objeto dado o un evento con la etiqueta léxica apropiada, y cómo funciona el proceso de recuperación léxica dentro del contexto de la producción de expresiones verbales.

El estudio de la producción se ha hecho basándose en observaciones, no en experimentos, dado que esto es inaccesible, por lo que hoy se ha hecho más fácil abordarlo observando los errores que cometen los hablantes. Estos estudios han llevado a la teorización alrededor de dos componentes de la producción: el "qué decir" y el "cómo decir". Respecto al primero ("qué decir"), este componente involucra la planeación a un nivel pre-lingüístico, donde el hablante decide, acorde a la representación que tiene de su co-hablante y del mensaje que quiere comunicar, cómo elegir y ordenar su "qué decir".

Garrett(1982)[19] analizó errores en hablantes normales y concluyó que hay dos niveles de planeación: el funcional, en el que el elemento se funda en una forma abstracta, alcanzando solo la información gramatical y semántica, y el nivel posicional, donde los morfemas gramaticales y fonológicos son insertados y donde se provee el orden de la producción.

Batterworth(1980)[7] analizó pausas en hablantes espontáneos y afirma que la frecuencia de las pausas depende de la complejidad de la planeación del contenido semántico del discurso, la programación semántica no es inconsciente, está en parte controlada; también asegura que la frecuencia de las pausas no se afecta por la complejidad sintáctica, lo cual indica que la programación sintáctica involucra procesos automáticos, por último, dice que la frecuencia de las pausas es alta durante la selección léxica de palabras las cuales tienen un alto grado de incertidumbre.

El resultado de estas investigaciones puede aplicarse a la perspectiva computacional en términos de heurísticas.

El segundo componente ("cómo decir") se refiere a la traducción del lexicón mental en una expresión verbal para la que se requiere haber tomado varias decisiones: la elección de la palabra, el orden de la enunciación y el acto de habla.

Para un contenido semántico dado hay muchas realizaciones lingüísticas. Hay varios parámetros que tomar en consideración:

- El efecto automático al usar una palabra que tenemos en la mente, esto se relaciona a la frecuencia de uso.
- El efecto preparatorio, que selecciona una palabra pre-activada por el contexto, por ejemplo, seleccionamos con mayor facilidad la palabra *cliente* en el contexto de un bufete de abogados, que la palabra *paciente*.
- La elección del orden de la enunciación y del acto de habla determinado, esto es, si es una afirmación o una pregunta, si usar o no modalidades, como "Creo que..." y el orden de la argumentación.

La psicolingüística no ha llegado a una conclusión de si la recuperación de información es un proceso serial o paralelo, esto es, no sabe si el reconocimiento de las palabras es un proceso controlado o bien tenemos un acceso directo y automático a los elementos léxicos. Los experimentos con estímulos visuales indican que el contexto no rige la existencia del nivel automático de reconocimiento de palabras, que es indiferente a la información contextual. Marslen y Wilson no están de acuerdo con esto en lo que toca a los actos de habla. Además, el acceso a distintos componentes de la expresión (por ejemplo a los niveles fonológico, léxico y sintáctico) no se hace en forma serial y dirigida, en la que cada componente es autónomo, recibiendo información del nivel superior y dando información al siguiente nivel más abajo. En lugar de esto, los componentes son accedidos en paralelo, con una parte desarrollada en un modo dirigido por el concepto, con guía de análisis semántico, por ejemplo, análisis fonológico y sintáctico.

Para Rumelhart, et. al. (1986)[40] tanto el tratamiento automático como el tratamiento controlado son necesarios. Distinguen como procesos automáticos a los que tardan menos de 0.25 a 0.5 segundos. Éstos, dicen, son paralelos y su modelo tendrá que ser paralelo. Los más lentos, tienen un componente serial y su modelo será de modelo de procesamiento secuencial.

Como vemos, el lexicón mental es esencialmente subjetivo (producto de la experiencia individual) y se maneja por el principio de la eficiencia, más que por el de economía (con trayectorias de acceso múltiple que permiten gran flexibilidad y la coherencia de su buen funcionamiento), por lo que aún ningún sistema propuesto por los psicolingüistas representa una solución aplicable y completamente satisfactoria computacionalmente.

La forma en que los lexicones están estructurados y organizados tiene un alto impacto en las formas en que deben ser accedidos. Una de las teorías que abordan este aspecto es la semántica componencial de Katz y Fodor. Esta teoría

fue inaugurada por Hjelmslev, y más adelante desarrollada por Katz y Fodor. Según ésta, una palabra se identifica por:

- marcadores sintácticos
- marcadores semánticos
- diferenciadores semánticos
- delimitación sintáctica o semántica contextual.

El significado de un ítem léxico se analiza en términos de átomos de sentidos (elementos terminales de la estructura de red).

Otros trabajos han añadido la noción de prototipo que determina a las palabras más típicas en una familia de palabras, por ejemplo "manzana" y "naranja" en la familia "frutas", también se refiere a encontrar los atributos prototípicos de la palabra, como por ejemplo, de "casa" serían: extensión del terreno, número de pisos, número de habitaciones, funcionalidad, ubicación, precio, etc. Asimismo, estos atributos pueden tener valores prototípicos, como número de pisos, en "casa" es prototípicamente entre 1 y 4 (si son más estaríamos hablando de otro tipo de construcción).

Otra teoría que se refiere a la estructuración y organización de los lexicones es la desarrollada por Miller y Johnson-Laird (1976)[33]. Ésta consiste en definir el significado de una palabra por los procedimientos involucrados en su uso. La palabra no se considera una entidad estática (un conjunto de características o proposiciones), sino como una subrutina la cual es llamada por el programa en turno que esté "ejecutándose", ya se encuentre en su fase de producción o reconocimiento. El sentido de "oración" se relaciona al de "programa de computadora". El modelo toma en cuenta la producción y el reconocimiento usando el mismo conjunto de reglas, lo cual es ciertamente una ventaja para los sistemas de inteligencia artificial. Sin embargo esto no los hace más apegados a la

investigación psicolingüística, al contrario. Un ejemplo es el siguiente: la expresión: "¿trajo Lucy el postre?" según Miller y Johnson-Laird, generaría un programa cuyos pasos son los siguientes:

- encontrar en memoria el episodio donde $F(x, y)$ existe.
- asignar a F el valor "traer", asignar a x "Lucy" y asignar a y "postre".
- Si la descripción se encuentra, la respuesta es "Sí"; si nada se encuentra, la respuesta es "No lo sé"; si alguna información contradice la descripción, la respuesta es "No".

Desde el punto de vista léxico, "traer" puede llamar a una subrutina la cual consiste en:

- encontrar en el dominio a consideración un evento e donde $venir(e)$ exista.
- verificar si, durante e , $traer(x, y)$ exista.

En esta perspectiva una palabra no está nunca aislada y no es independiente del contexto, y su significado se ve como una subrutina dentro de un programa. Sin embargo se ha dicho muy poco sobre la semántica y la sintaxis de las palabras.

La aplicación de la semántica léxica en la lingüística actualmente está relacionada principalmente con: la estructura de argumento, los roles temáticos, las restricciones selectivas, la semántica léxica, las relaciones esenciales, las ontologías, la estructura conceptual léxica, la estructura de eventos y la estructura *qualia*.

La **estructura de argumento** es una estructura que representa conceptos a través de predicados y un número de argumentos. Los argumentos representan los

elementos que necesariamente están involucrados en la acción o el estado descrito por el predicado. Se realiza de la siguiente manera:

Dar: 3 Esto indica que "Dar" necesita de tres argumentos X, Y, Z
(el_que_da, el_que_recibe, lo_que_se_da)

Pensar: 2

Dormir: 1

Etc.

Los **roles temáticos** se han usado en la lingüística aplicada con un alto grado de sofisticación. Los aspectos más relevantes a la semántica léxica computacional son: la asignación de rol temático, los roles temáticos de organización jerárquica y proto-roles temáticos. Los roles temáticos más prominentes son: el agente (el participante designado por el predicado para hacer o causar la acción), el paciente (participante afectado por la acción), el tema (el participante que cambia de lugar, condición o estado o que está en determinado estado o posición), el experimentador (el participante que se informa de algo o que experimenta un estado psicológico por lo expresado en el predicado), la fuente (objeto desde el cual ocurre el movimiento), meta (objeto hacia el que el movimiento es dirigido), lugar (sitio en que la acción o estado descrito por el predicado tiene lugar) y beneficiario (el participante que se beneficia de la acción expresada por el verbo). Los roles temáticos pueden asociarse en forma jerárquica y en ocasiones puede resultar de interés descomponer varios roles, que individualmente no son de mucho interés práctico, en fragmentos de significado formando "clusters".

Las **restricciones selectivas** introducen un criterio más pragmático y dependiente del dominio en el contenido de los argumentos. Las restricciones son muy usadas en las aplicaciones NLP⁷ y están frecuentemente basadas en las

⁷ Natural Language Processing.

técnicas de representación del conocimiento de la Inteligencia Artificial. El principio de esta técnica es asociar cada argumento de un predicado a una lista de características F_i que expresan restricciones en la naturaleza semántica de los argumentos. F_i podría ser una simple característica, como "humano", "animal", etc., mientras que $(F_1 \wedge F_2 \dots \wedge F_n)$ es una conjunción de restricciones, y $(F_1 \vee F_2 \dots \vee F_3)$ es cuando alguna restricción debe cumplirse. También se hacen combinaciones de estas formas. Las restricciones selectivas se basan frecuentemente en bases empíricas, en los últimos años se han tratado de definir principios para formarlas, una de estas orientaciones organiza los tipos semánticos usados como restricciones selectivas a través de un árbol de tipos. Estos tipos están estructurados a través de una relación tipo-subtipo, por ejemplo, un humano es un subtipo de una entidad animada.

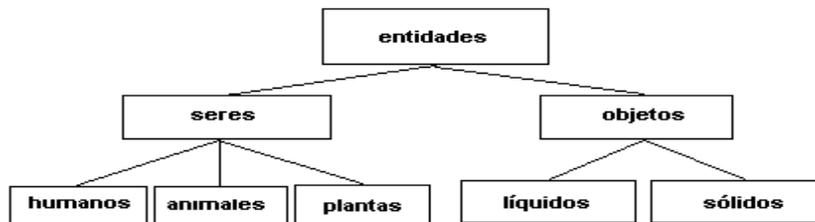


Figura 5: Relación tipo-subtipo.

Las **relaciones semánticas léxicas** permiten hacer un análisis de las palabras en función a la relación entre las palabras. Las relaciones de congruencia son: identidad (cuando las entidades E_1 y E_2 son iguales: $E_1 = E_2$), de inclusión (E_2 está incluida en E_1), la superposición (cuando E_1 y E_2 tienen una intersección no vacía, pero no son subconjuntos alguno del otro) y la disyunción (no tienen elementos en común E_1 y E_2). También pueden estar relacionadas

jerárquicamente, las relaciones de este tipo son: la taxonomía (*'is_a_kind_of'*, *'is_a_way_of'*), las meronimias (*'has_a'*, *'is_part_of'*) y las series proposicionales.

También hay relaciones no-jerárquicas como la sinonimia, la antonimia y los opuestos.

Una **ontología** es un sistema formal que permite la representación de un dominio dado a través de sus elementos básicos, los distintos conceptos y sus realizaciones lingüísticas relacionadas. Una mirada extensa hacia el conocimiento ontológico puede incluir varias formas de conocimiento enciclopédico sobre el dominio, a la vez que conocimiento de sentido común, y conocimiento retórico y metafórico, así como expresiones. Las descripciones ontológicas son usadas en múltiples situaciones y clases de aplicaciones. En términos de la aplicación, algunos sistemas incorporan conocimiento ontológico para mejorar su calidad y generalidad. Este tipo de conocimiento puede ser crucial para interfaces de usuario inteligentes de lenguaje natural y para la recuperación automática de documentos, aplicándose mucho a las redes semánticas, a los grafos conceptuales, a la estructura conceptual léxica, a los léxicos generativos y a los sistemas de administración de escenarios, *scripts* y discursos.

La **estructura conceptual léxica** permite establecer una correspondencia basada en la noción del concepto primitivo entre el lenguaje externo y el lenguaje interno. Los conceptos básicos son léxicos y el significado de una oración se construye a través de los significados léxicos. Es así como cada constituyente léxico tiene una representación conceptual, cada categoría conceptual codifica una o más unidades lingüísticas, además de que establecen la diferencia entre "tipo" y "objeto"; las categorías son también cuantificables. La estructura conceptual de un elemento léxico es una entidad con cero o más argumentos.

Una **estructura de evento** permite identificar un tipo particular de evento asociado a la palabra o frase, por ejemplo, el logro, el estado. Básicamente una palabra o frase denota un estado, un proceso o una transición.

Las **estructuras de herencia Qualia** son estructuras que proveen los atributos esenciales para un objeto, distribuido entre 4 aspectos del significado de la palabra, y capturada en alguno de los siguientes roles: el rol formal (el que lo distingue en otros dominios más amplios), el rol constitutivo (que da la relación entre el elemento léxico y sus constituyentes), el rol “tético” (el que define propósito y meta) y el rol de agente (que trae información diversa). Estas estructuras forman parte de una teoría que busca sustentar la generación de un lexicón altamente organizado y estructurado. Pustejovsky(1991)[36] es el autor de dicha teoría y de dicho lexicón, se basa en la idea de que esta teoría del significado léxico puede determinar el significado esencial de los elementos léxicos.

3 Selección automática de primitivas semánticas

3.1 Conceptos básicos

El metalenguaje semántico natural (NSM -*Natural Semantic Metalanguage*-) es una teoría desarrollada por Anna Wierzbicka y más tarde también por Cliff Goddard (1999)[20], que incluye ciertos principios y un método para construir un vocabulario de primitivas semánticas. Uno de los principios es la paráfrasis reductiva. La paráfrasis reductiva (*reductive paraphrase*) consiste en describir un significado complejo en términos de otros más simples, esto es, para consignar el significado de una palabra compleja (semánticamente hablando), se dará una paráfrasis formada por otras palabras más "simples" y más fáciles de entender que la original, por ejemplo:

X ama a Y =

X frecuentemente piensa en Y

X piensa cosas buenas sobre Y

X quiere hacer cosas buenas por Y

X quiere que cosas buenas le sucedan a Y

Cuando X piensa en Y, X frecuentemente quiere estar con Y

Cuando X piensa en Y, X frecuentemente siente algo bueno

Según esta teoría, cada lenguaje tiene un conjunto de primitivas que es irreductible y del cual hacen uso los hablantes para entender todo pensamiento complejo. Éste conjunto tiene la característica de ser el mismo en todos los idiomas, pues es un reflejo del pensamiento humano. Cada concepto específico a una cultura puede traducirse a una configuración de primitivas semánticas

Las primitivas semánticas y sus principios de combinación serían como un mini-lenguaje con la misma expresividad que el lenguaje natural. El modelo actual de primitivas semánticas de Anna Wierzbicka (1996)[47]⁸ es el siguiente:

Sustantivos	YO, TÚ, ALGUIEN, GENTE, ALGO, CUERPO
Determinadores	ESTO, LO MISMO, OTRO
Cuantificadores	UNO, DOS, ALGO, TODO, MUCHO/MUY
Evaluadores	BUENO, MALO
Descriptorios	GRANDE, PEQUEÑO, (LARGO)
Predicados mentales	PENSAR, SABER, QUERER, SENTIR, VER, ESCUCHAR
Habla (discurso)	DECIR, PALABRA, CIERTO
Acciones, eventos y movimientos	HACER, PASAR, MOVER, (TOCAR)
Existencia y posesión	HAY, TENER
Vida y muerte	VIVIR, MORIR
Tiempo	CUANDO/TIEMPO, AHORA, INICIO, DESPUÉS, UN TIEMPO GRANDE, UN TIEMPO CORTO, POR ALGÚN TIEMPO, (MOMENTO)
Espacio	DONDE/LUGAR, AQUÍ, SOBRE, BAJO, LEJOS, CERCA, LADO, DENTRO
Lógica	NO, MAY BE, PORQUE, SI
Intensificador, aumentador	TODO, MÁS
Taxonomía, "partonomía"⁹	TIPO DE, PARTE DE
Similaridad	COMO

Tabla 1. Primitivas semánticas de Wierzbicka.

⁸ Esta tabla tiene las traducciones de las palabras que esta investigadora determinó a partir del idioma inglés. Se realizó una traducción al español con las imprecisiones que esto conyeva.

Como se puede ver en esta tabla, Wierzbicka ha encontrado aproximadamente 60 primitivas semánticas.

Algunas observaciones que se desprenden respecto de esta lista de primitivas (mismas que la autora ha reconocido) son:

- Una lista de primitivas no es suficiente pues hay términos que también tienen múltiples sentidos.
- Una caracterización completa incluye los "contextos canónicos" (conjunto de oraciones o partes de oraciones ejemplificando contextos gramaticales para cada primitiva).
- Las primitivas pueden ser frasemas, por ejemplo: "un tiempo grande".
- Las primitivas no siempre son morfológicamente simples (en inglés: *someone, inside*)
- Pueden ser alomorfos (por ejemplo, en inglés *thing-something*)

Como ya se mencionó en la sección 2.2 los *scripts* son convenciones locales de discurso que se expresan a través del metalenguaje de primitivas semánticas inventado por Wierzbicka, por ejemplo, en japonés:

Las personas piensan que:

Si algo malo pasó porque yo hice algo.

Yo tengo que decir algo a alguien porque yo hice algo.

"Yo siento algo malo".

Yo tengo que hacer algo debido a esto.

Este *script* representa la necesidad de disculparse que está presente en la cultura japonesa y la cual es distinta en el mundo occidental.

⁹ En inglés *partonomy*, aludiendo a la acción de dividir en partes algo.

Por otro lado, Apresjan llegó a la conclusión de que sería de gran utilidad usar un conjunto restringido de palabras de un lenguaje para construir las definiciones de las otras palabras de ese lenguaje, pero en su opinión este conjunto no sería muy pequeño. Como hemos visto en la sección 2.6.1 el vocabulario definidor del *Longman Dictionary* se compone de más de 2000 palabras.

3.2 Algoritmo principal

Nuestra solución utiliza los métodos de la lexicografía computacional (Saint-Dizier and Viegas).[41]

Específicamente, representamos el diccionario como un grafo dirigido. La idea de representar el diccionario como un grafo (una red semántica), no es nueva, es una idea desarrollada por autores como Evens[13], y Fellbaum[14]. Kozima y Furugori [27] también analizan una red semántica, en este caso, para saber qué palabras se "activan" empezando de alguna palabra determinada.

En nuestro grafo, los vértices son las palabras que se mencionan en el diccionario, tanto las palabras que se definen como entradas, como las que se usan en las definiciones. La misma palabra puede ocurrir tanto como una entrada como varias veces en las definiciones, pero se cuenta como el mismo vértice.

Hay diferentes maneras de considerar que dos ocurrencias (*tokens* en inglés) son "la misma palabra" (*type* en inglés): como cadenas de letras, o por la forma base, o por la raíz, o por el significado específico, etc. En la sección 0 se dan más detalles sobre los dos métodos que aplicamos (Grafo 1, por forma base, y Grafo 2, por significado).

Definidos los vértices, las flechas o arcos del grafo se definen como sigue: la flecha desde la palabra v_1 hasta la palabra v_2 significa que en la definición de la palabra v_1 se usa la palabra v_2 . Las palabras que no son entradas no tienen flechas

salientes, mientras que las que no se usan en las definiciones de otras palabras, no tienen flechas entrantes. Por ejemplo:

Una entrada del diccionario Anaya es la palabra "prolepsis", esta palabra tiene dos sentidos:

Prolepsis.- De gr. *pro-lepsis*:percepción. I. Figura retórica consistente en adelantarse a las posibles objeciones del interlocutor. II. Construcción gramatical consistente en colocar un elemento en una unidad sintáctica anterior a la que le corresponde lógicamente.

Esta palabra es entrada, tiene flechas salientes (las palabras de su definición, como se muestra en la figura 6), pero no hay flechas entrantes hacia ella, pues el resto de las palabras del diccionario no la usan en sus definiciones.

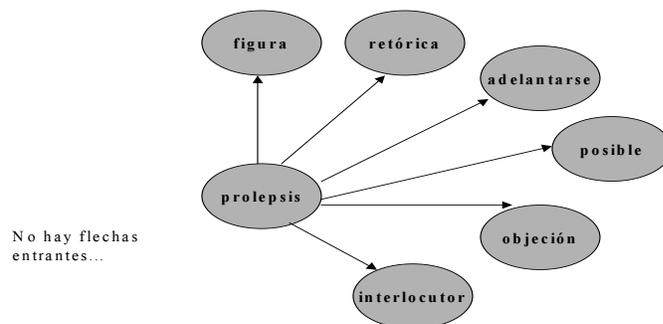


Figura 6. Uno de los sentidos de la palabra "prolepsis".

"Metempsicosis" es otra de las palabras que son entradas del diccionario, pero que no se emplean en las definiciones de otras palabras. En cambio, otras palabras, como "cobla", tienen en su definición palabras como "Cataluña", que no son entradas del diccionario, como se muestra a continuación:

Cobla.- Del lat. *copula*: unión. I. En Cataluña, grupo de músicos que tocan sardanas.

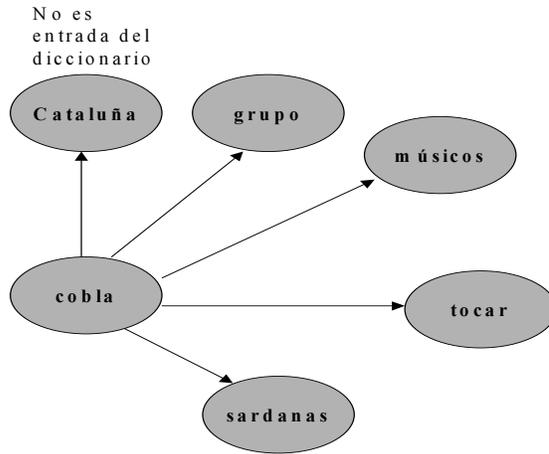


Figura 7. Definición de "cobla".

Otras palabras de este tipo (que no son entradas del diccionario aunque aparecen en las definiciones) son: "Grecia", que aparece en la definición de la palabra "griego", "María" que aparece en la definición de "magnificat", "Méjico" que aparece en la definición de "corrido", etc. Como vemos se trata de nombres propios de lugares o personajes, dado que el diccionario Anaya no es enciclopédico.

3.3 Modelo matemático

Desde el punto de vista matemático, el problema y su solución son los siguientes.

Sea $G = \{V, F\}$ un grafo dirigido definido por los conjuntos V de N vértices y $F \subseteq V \times V$ de flechas. Un ciclo en este grafo, lo entenderemos como un ciclo dirigido.

Digamos que un subconjunto $P \subseteq V$ es un subconjunto *definidor* si cualquier ciclo en el grafo G contiene un vértice de P . O sea, si el grafo $G' = \{V', F'\}$ que se obtiene del G eliminando los vértices que pertenecen al P –es decir,

$V' = V \setminus P$ y $F' = F \cap (V' \times V')$ – no tiene ciclos. Llamaremos los vértices $p \in P$ los definidores.

Digamos que un subconjunto definidor $P \subseteq V$ es *mínimo* si ningún subconjunto $P' \subset P$ es definidor. O sea, si la eliminación de cualquier elemento de P produce un subconjunto no definidor. Es decir, para cada vértice $p \in P$, existe un ciclo en G que contiene p y no contiene ningún otro vértice de P .

El conjunto *definidor mínimo* no tiene que ser el más pequeño (el que está contenido en todos los subconjuntos definidores; tal subconjunto usualmente no existe) ni siquiera del tamaño más pequeño posible: como es muy fácil mostrar con ejemplos y como también será claro en nuestro algoritmo, en el mismo grafo pueden existir muchos conjuntos definidores mínimos de tamaños diferentes.

El algoritmo que aquí presentamos resuelve el siguiente problema: dado un grafo dirigido G , encontrar un subconjunto definidor P mínimo, aunque no sea del tamaño más pequeño.

El algoritmo encuentra una de múltiples variantes posibles de un subconjunto definidor mínimo. La selección de la variante se puede controlar con un ordenamiento $\sigma: \{1, \dots, N\} \leftrightarrow \{1, \dots, N\}$ que ordena los números de 1 a N en una secuencia $\sigma(1), \dots, \sigma(N)$. Por ejemplo: 3, 5, 2, 4, 1 es un ordenamiento para $N = 5$.

Los conjuntos P generados basándose en diferentes ordenamientos σ son usualmente diferentes.

Como se observará en el algoritmo, el sentido del ordenamiento es el siguiente: los primeros vértices tienden a ser no definidores y los últimos tienden a entrar en P . Específicamente, el primer vértice en el ordenamiento siempre es no definidor (si no tiene un lazo).

Ahora bien, dado G y σ , el algoritmo funciona como sigue. Se construye, paso a paso, un subgrafo $G' \subseteq G$ sin ciclos. Inicialmente está vacío. Se construye, insertándole uno por uno (en el orden dado σ), los vértices de G con sus relaciones hacia los vértices ya insertados. En cada paso, G' se mantiene sin ciclos: si el vértice a insertar le causa ciclos, se considera un definidor y no se inserta en G' . Al terminar el proceso, se tiene el conjunto definidor P , que por su construcción es mínimo (porque cada $p \in P$ produce ciclos incluso en un subconjunto del G').

El paso computacionalmente más costoso del algoritmo es la verificación de que el nuevo vértice no le causaría ciclos al subconjunto G' . Por esto, para cada vértice $v_i \in V$ el algoritmo mantiene un conjunto A_i de vértices accesibles en G' desde v_i , diciéndose del vértice u que es accesible desde v si existe un camino dirigido en el grafo desde v hasta u . El algoritmo se basa en el hecho de que es una relación transitiva: si a es accesible desde b y b desde c , entonces a es accesible desde c .

Formar el conjunto de los definidores $P = \emptyset$ y de los no definidores $V' = \emptyset$.

Paso previo. Con el fin de disminuir el conjunto de datos a procesar se puede llevar a cabo la depuración del grafo, véase más adelante el algoritmo B. Éste consiste en agregar elementos al conjunto P y eliminar los vértices (y sus relaciones) del conjunto G .

1. Para $i = 1, \dots, N$ repetir:
2. Seleccionar $\sigma(i)$ -ésimo vértice $v \in \mathbf{V}$.
3. Verificar si v se puede agregar al grafo sin provocar ciclos. Para esto:
4. Para cada flecha $(v \rightarrow u) \in \mathbf{F}$ que va desde v tal que $u \in \mathbf{V}'$ repetir:
5. Para cada flecha $(v \leftarrow w) \in \mathbf{F}$ que va hasta v tal que $w \in \mathbf{V}'$ repetir:
6. Verificar que $w \notin \mathbf{A}_u$.
7. Si el paso 6 falla, agregar v a \mathbf{P} .
8. En caso contrario, agregarlo a \mathbf{G}' . Para esto:
9. Agregar v a \mathbf{V}' .
10. Formar $\mathbf{A}_v = \cup \mathbf{A}_u$, donde la unión se forma por todas las flechas $(v \rightarrow u) \in \mathbf{F}$ que van desde v tales que $u \in \mathbf{V}'$.
11. Para cada flecha $(v \leftarrow w) \in \mathbf{F}$ que va en v tal que $w \in \mathbf{V}'$ repetir:
12. Para cada vértice $q \in \mathbf{V}'$ tal que $q = w$ ó $w \in \mathbf{A}_q$ repetir:
13. Agregar $\{v\} \cup \mathbf{A}_v$ a \mathbf{A}_q .

Figura 1. Algoritmo A.

Después del paso previo, se supone que \mathbf{V} , \mathbf{F} y \mathbf{N} están definidos por el grafo \mathbf{G} ya depurado, es decir, posiblemente sean menores que el original. También el algoritmo mantiene el conjunto \mathbf{V}' de los vértices ya incluidas en el subconjunto \mathbf{G}' y el conjunto \mathbf{P} (se pueden considerar como unas coloraciones de los vértices del conjunto \mathbf{G} pues no se intersectan). Al terminar el algoritmo, \mathbf{P} será un conjunto definidor mínimo. Véase el algoritmo detallado en la figura 1.

El algoritmo tiene complejidad cúbica, siendo las operaciones que se repiten el mayor número de veces los pasos 0 y 0. Dado el gran tamaño del diccionario (30 mil vértices en nuestro caso), dos precauciones se deben tomar para mantener su tiempo de espera en los límites razonables:

- Implementar eficientemente la operación más crítica y
- Disminuir el tamaño de los datos a procesar.

En cuanto a la implementación eficiente, en una computadora PC nosotros representamos los conjuntos A_i como una matriz de $N \times N$ bits empacados en $N \times \left(\frac{N}{32} + 1\right)$ números enteros de 32 bits. La operación de unión de dos conjuntos (dos renglones de esta matriz) se implementa a través de la disyunción lógica (OR) de los números enteros, lo que es 32 veces más rápido que operar sobre cada elemento independientemente.

También en la misma PC Pentium II, probamos la implementación a través de las unidades de 64 bits usando la tecnología MMX, pero no observamos incremento considerable en la rapidez en comparación con las operaciones en 32 bits.

En cuanto a disminuir el tamaño de los datos a procesar, se pueden observar dos hechos:

1) Los vértices v con un lazo $(v \rightarrow v) \in F$ tienen que estar en P pues no pueden estar en $G^?$. Nótese que la prueba en el paso 0 falla para tales vértices. Entonces, estos vértices –aunque son pocos– se pueden de antemano agregar a P y quitar (con todas sus relaciones) de G para fines del procesamiento posterior.

2) Los vértices que no tienen ninguna flecha entrante o ninguna flecha saliente no pueden estar en P (recuerde que P es un subconjunto definidor mínimo) pues para tal vértice v , si Q es un conjunto definidor entonces $Q \setminus \{v\}$ también lo es. Nótese que la prueba en el paso 0 no se ejecuta para tales vértices y de este modo no puede fallar. Entonces, estos vértices –y en nuestro caso son muchísimos– se pueden de antemano quitar (con todas sus relaciones) de G considerándolos no definidores.

1. Para cada vértice con lazo, repetir:
2. Agregar el vértice al P y eliminarlo del G junto con sus flechas.
Repetir:
3. Para cada vértice que no tiene salientes o entrantes repetir:
4. Eliminarlo de G junto con sus flechas, considerándolo no definidor.

Figura 2. Algoritmo B.

La eliminación de un vértice puede hacer que otros vértices pierdan todas sus flechas entrantes o salientes. Entonces, estas dos observaciones sugieren el algoritmo de depuración mencionado en el Paso previo del algoritmo descrito más arriba, véase la figura 2.

La eliminación de un vértice se puede implementar como una coloración. El grafo G que se obtiene después de la depuración, satisface las siguientes condiciones:

1. No tiene lazos.

2. Cada vértice es transitivo, es decir, tiene flechas tanto entrantes como salientes. En tal grafo, para cada vértice v existe un conjunto definidor mínimo P tal que $v \notin P$. Más generalmente, para cada conjunto compatible $Q \subseteq V$ (diciéndose que el conjunto Q es compatible si no existen ciclos en G que contengan únicamente los vértices de Q) existe un conjunto definidor mínimo P tal que $Q \cap P = \emptyset$. Esto se comprueba por la aplicación del algoritmo A con un ordenamiento σ que empieza con los elementos de Q .

Entonces, en tal grafo la propiedad de ser un definidor no es propia del vértice sino sólo es relativa a la selección de un conjunto P . Es decir, ningún

vértice –salvo los vértices con lazos– es un definidor “por sí mismo”, o bien, que entra en cualquier conjunto definidor.

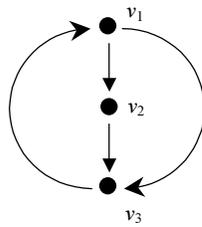


Figura 3. Contraejemplo.

Lo contrario no es cierto: en el grafo depurado con el algoritmo B todavía pueden existir los vértices que no entran en ningún conjunto definidor del tamaño mínimo, es decir, se pueden eliminar de cualquier conjunto definidor. Un ejemplo de tal situación –el vértice v_2 – se muestra en la figura 3. Nosotros no tenemos ningún algoritmo rápido para detectar tales vértices. Sin embargo, nuestros experimentos con el diccionario real mostraron que en este diccionario los vértices de este tipo, si existen, no constituyen más del 20% del grafo depurado, pues encontramos los conjuntos definidores mínimos cuya unión cubre un 80% del grafo. Por lo tanto, la detección previa de tales vértices no contribuiría significativamente en la rapidez del algoritmo.

3.4 Implementación del algoritmo

El sistema por medio del cual procesamos el diccionario está formado de 3 módulos o programas y una librería, los cuales son:

- a) el módulo **graph**,
- b) el módulo **circle**,

- c) el módulo **vis**, y
- d) la librería **circle**.

En términos generales es módulo **graph** controla la lectura, depuración y visualización del grafo. También se encarga de ejecutar alguno de los 4 métodos o estrategias explicadas en la sección 4. En lo referente a la lectura, depuración y visualización, **graph** llama a métodos que se encuentran desarrollados en el módulo **circle**, por lo que éste módulo no se ejecuta directamente por el usuario. El módulo **vis**, en cambio, es un módulo que ejecuta el usuario para realizar una verificación de las primitivas seleccionadas, o bien, para crear un archivo en el que se visualice el conjunto de primitivas elegido de cualquiera de las siguientes formas: con los datos básicos de cada primitiva o con los datos básicos más uno de los ciclos más cortos que rompe cada primitiva en la red depurada. La librería **circle** declara algunas de las clases y métodos usados por los módulos **graph**, **circle** y **vis**. A continuación se detallará esta breve descripción de la implementación del algoritmo.

3.4.1 Módulo **graph**

Este módulo es el que controla la depuración del grafo de entrada y la elección de un subconjunto de palabras que cortan los ciclos en el diccionario y nos permite obtener lo que llamaremos un "vocabulario definidor" mínimo. Es así como **graph**, lee el grafo y aplica el algoritmo B llamando a un método del módulo **circle**. Asimismo, **graph** es el módulo que contiene en sus métodos el algoritmo A. Ambos algoritmos se mostraron en la sección 3.3. En la siguiente figura aparecen las partes que forman al módulo **graph**:

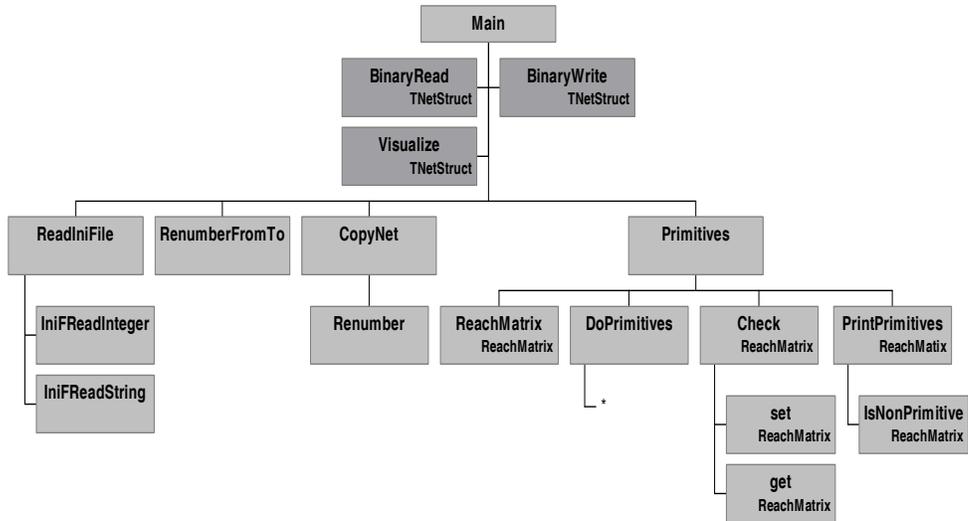


Figura 8. Módulo Graph (primera parte).¹⁰

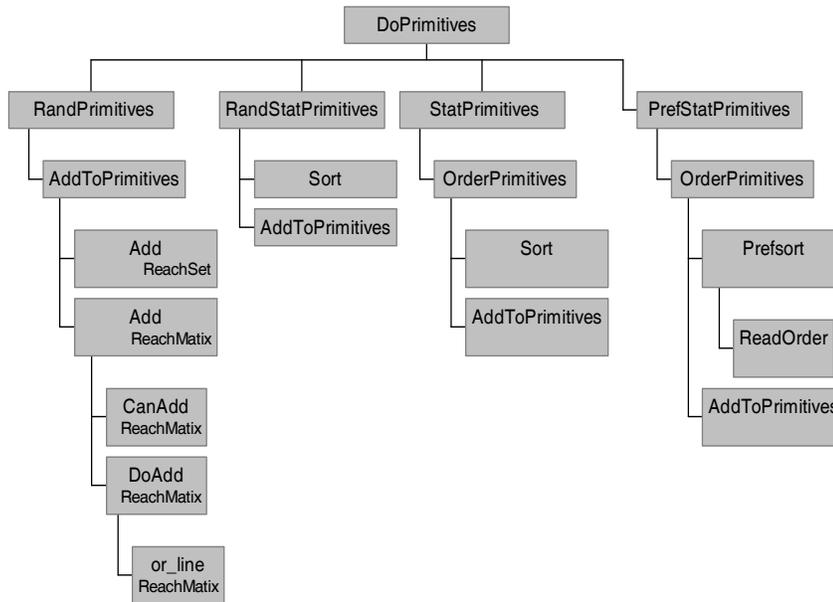


Figura 9. Módulo Graph (continuación).

¹⁰ Los bloques de color más oscuro de estos diagramas de bloques representan métodos desarrollados en otros módulos.

El módulo **graph** puede recibir dos parámetros. El parámetro *c* indica que el programa tomará como grafo de entrada el contenido en el archivo **net.txt** el cual está en formato binario, para obtener como salida el grafo depurado **net1.txt**, el cual también está en formato binario, conforme al algoritmo B. En **net1.txt** se han eliminado todas aquellas palabras que no presentan ningún ciclo en su definición, dado que estas palabras, según nuestra aproximación, no serán consideradas como definidoras. Si no usamos la opción *c* entonces el archivo de entrada será **net1.txt** por defecto. El parámetro *v* sirve para crear el archivo **net1.vis**, el cual contiene las 10,358 entradas del diccionario que forman el grafo depurado, con los nodos a los que apuntan sus flechas de entrada y de salida, en un formato adecuado para su lectura, éstas palabras crean conjuntos de nodos altamente interconectados. El programa **graph.exe** sin parámetros, creará otro archivo, llamado **primit.out**, el cual contendrá la lista de primitivas encontradas según alguno de los cuatro algoritmos o estrategias de ordenamiento.

La forma como trabaja el algoritmo A se explica a continuación. Como primer paso **graph** lee el archivo de inicio **graph.ini**, el cual contiene una cadena que le indica cuál método de selección de primitivas usar, por ejemplo: Algorithm = 2, lo cual indicaría que se trata del método 2, por frecuencias. Enseguida, si la opción elegida es *c*, es decir, *clean*, se hace una lectura binaria de la red **net** a través del método **BinaryRead** de la clase **TNetStruct**. Posteriormente ejecuta la función **Cleanup** que elimina a las entradas que no conducen a ciclos, ya que hemos considerado a éstas, por dicha característica, como no-primitivas. El siguiente paso del algoritmo es la inicialización de la red, es decir, la propiedad "*marcado*" de los nodos se establece como **false** para todos ellos, y reenumera para después copiar la parte de la red con la que va a trabajar al grafo **net1**, convirtiéndolo también a formato binario.

Este proceso de copiar una parte de la red **net** a **net1** no se realiza si se omite el parámetro *c* y por lo tanto se toma simplemente, la red **net1** como la

proveedora de posibles conjuntos de primitivas semánticas. Cuando el parámetro es v , el método para visualizar la red **net1** es **Visualize (módulo Circle)**.

La función que obtiene las primitivas se llama **Primitives** y tiene como parámetros a **net1** y al archivo de salida **primit.out**.

Los métodos a elegir son los siguientes:

Método 1. Genera un número aleatorio entre los N nodos de la red, de manera que cada uno de los nodos sea evaluado como candidato a ser primitiva.

Método 2. Ordenamos los vértices por el número de las flechas entrantes de menor a mayor y en ese orden se procesan los nodos para provocar que los nodos con mayor frecuencia sean los que queden en el conjunto de primitivas.

Método 3. Calcula la frecuencia máxima y genera un número aleatorio, pero con las probabilidades en función inversa a las frecuencias.

Método 4. Para este método genera 20 diferentes conjuntos definidores mínimos P_i con el método 1, asociando así con cada vértice un número entre 20 y 0. Ordena primero los vértices que nunca entraron en los P_i usando sus frecuencias, como en el método 2, y después los que entraron, en el orden inverso al número de los P_i en que entró, es decir, desde 1 hasta 20.

3.4.2 Módulo circle

En éste módulo se desarrollan los métodos para hacer una escritura y lectura binaria sobre cada nodo y sobre la estructura de la red. La escritura binaria de cada nodo se realiza a través del método **BinaryWrite** de la clase **TNode**, por medio del cual se guarda la información esencial de cada nodo para la posterior depuración del grafo y para su procesamiento. El método **BinaryRead** de la clase

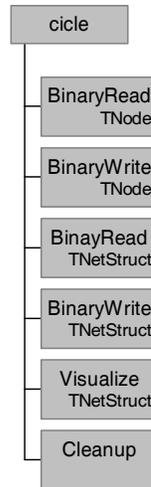


Figura 10. Módulo Circle.

TNode sirve para hacer una lectura desde el archivo **net1.txt** ? de las características de cada nodo. El método **BinaryWrite** de la clase **TNetStruct** controla la escritura de los nodos en el archivo que contiene al grafo (depurado o no). El método **BinaryRead** de la clase **TNetStruct** abre el archivo que contiene el grafo y un nodo **TNode** del cual leerá sus características a través del método **BinaryRead** de la clase **TNode**. El método *cleanup* consiste en marcar los nodos que no tienen entradas o salidas, removiéndolos del grafo (**net**); después escribe en un archivo cuántos nodos fueron removidos y cuántos se conservaron. Posteriormente el módulo graph escribe el resultado en el archivo **net1.txt**.

3.4.3 Módulo vis

El módulo vis puede recibir los siguientes parámetros: *l* indica modo de lista, por lo en este modo, el módulo toma el archivo de primitivas dado por el usuario como segundo parámetro (en estos archivos sólo está una lista con los identificadores de las primitivas halladas) y el nombre del archivo en el que escribirá las primitivas conforme al modo de lista. En éste último escribe la

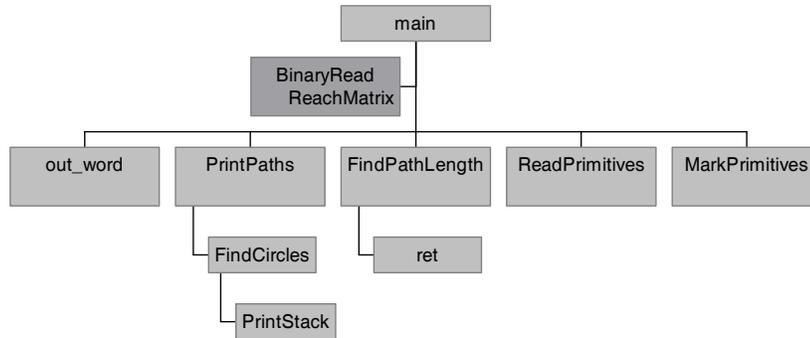


Figura 11. Módulo Vis.

frecuencia, el número de salidas, el número de entradas, la palabra y el identificador de cada primitiva. Si la opción elegida es *v*, el módulo tomará el archivo de entrada de primitivas dado por el usuario y lo procesará para obtener otro archivo que contendrá la longitud de uno de los ciclos más cortos, el número de salidas, el número de entradas, la palabra y el identificador de cada primitiva.

3.4.4 Librería circle

Esta librería contiene los atributos de la clase TNode, el constructor de dicha clase, la clase NetStruct y el constructor de esa clase.

3.4.5 Explicación del código.

A continuación describiremos las principales funciones que desempeñan los tres módulos que forman al sistema.

El módulo Graph define dos clases de datos, la clase ReachSet y la clase ReachMatrix.

ReachSet ¹¹
Métodos públicos ReachSet () ~ReachSet () GetItemsInContainer () Flush () Add (int n)
Métodos privados TArray<unsigned> s

Clase ReachSet.- Clase que respresenta a un contenedor de datos que se pueden agregar y quitar. Con este contenedor se realizaran las siguientes operaciones:

ReachSet () .- Crea un objeto de esta clase que es un conjunto **s** (0, 0, 1000) al que llamaremos contenedor.

~ReachSet () .- Destructor del objeto

GetItemsInContainer () .- Obtiene los datos del contenedor **s**.

Flush ().- Elimina los datos del contenedor **s**.

Add (int n).- Agrega el entero **n** al contenedor **s**, **s** representa un nodo que tiene una flecha entrante o una flecha saliente.

Clase ReachMatrix .- Clase que representa una matriz cuyos elementos se pueden agregar, quitar, revisar, etc. para finalmente identificar si un nodo es primitivo o no. En el caso de que lo sea, el identificador de este nodo se escribe en un archivo. A continuación se detallan los métodos públicos y privados que se aplican a esta clase, sus parámetros y su funcionamiento:

¹¹ Para describir las clases se utiliza en este trabajo la notación denominada *Unified Notation*

ReachMatrix
<p>Métodos públicos</p> <pre> ReachMatrix (unsigned size); ~ReachMatrix (); void Flush (); bool Add (unsigned n, const ReachSet & from_which, const ReachSet & which); bool Check () const; void PrintPrimitives (ostream & f) const; bool IsNonPrimitive (int i) const </pre>
<p>Métodos privados</p> <pre> bool CanAdd (unsigned n, const ReachSet & from_which, const ReachSet & which); void DoAdd (unsigned n, const ReachSet & from_which, const ReachSet & which); void or_line (unsigned i, unsigned j); // desde-hasta (from to) void or_line (char *pi, char *pj); // desde.hasta (from to) char *line (unsigned i) const; void set (unsigned i, unsigned j); void set (char *pi, unsigned j); bool get (unsigned i, unsigned j) const; bool get (char *pi, unsigned j) const; unsigned cols; unsigned lins; buftype *m; char *non_primitives; </pre>

Métodos públicos y sus parámetros:

1.1 ReachMatrix::ReachMatrix (unsigned size)

Método: ReachMatrix.

Clase: Reach Matrix.

Tipo: Constructor de la clase.

Parámetros: Recibe **size** que indica el número de nodos de la red.

Llamadas: no hay.

Constructor de la clase ReachMatrix, determina el número de líneas y columnas en función al tamaño de la red (**size**), es decir, al número de nodos de la red.

1.2 ReachMatrix::~~ReachMatrix (unsigned size)

Método: ~ReachMatrix.

Clase: Reach Matrix.

Tipo: Destrucción de la clase.

Parámetros: Recibe **size** que indica el número de nodos de la red.

Llamadas: no hay.

Destructor de la clase ReachMatrix

1.3 void ReachMatrix::~Flush ()

Método: Flush().

Clase: Reach Matrix

Tipo: void

Parámetros: no hay.

Llamadas: no hay

Quita una cantidad de bytes en un bloque de memoria que era para la matriz **m** y para el arreglo **non_primitives**.

1.4 bool ReachMatrix::~Check () const

Método: Check().

Clase: Reach Matrix

Tipo: booleano

Parámetros: no hay.

Llamadas: no hay

Crea y llena un archivo "matrix.out", numera los renglones de 0 a lins/100 y la llena con ceros y unos. Después revisa la matriz, y si encuentra errores regresa "falso"; necesita devolver "verdadero" para que el programa continúe.

1.5 void ReachMatrix::or_line (unsigned i, unsigned j)

Método: or_line

Clase: Reach Matrix

Tipo: void

Parámetros: Recibe dos enteros positivos **i** y **j**.

Llamadas: no hay

Forma alternativa de manejar la memoria

1.6 void ReachMatrix::or_line (char *pi, char *pj)

Método: or_line

Clase: Reach Matrix

Tipo: void

Parámetros: Recibe dos punteros a carácter **i** y **j**.

Llamadas: no hay

Forma alternativa de manejar la memoria cuando se trata de caracteres.

1.7 char *ReachMatrix::line (unsigned i) const

Método: line

Clase: Reach Matrix

Tipo: char

Parámetros: Recibe el entero positivo **i**

Llamadas: no hay

Recibe el identificador **i** de un nodo y devuelve un apuntador a carácter **line** con el valor **m+cols*i**

1.8 void ReachMatrix::set (unsigned i, unsigned j)

Método: set

Clase: Reach Matrix

Tipo: void

Parámetros: Recibe dos enteros positivos **i** y **j**.

Llamadas: no hay

Recibe dos variable **i**, **j**, llama a **line** y con su aputador a caracter y su número llama a la otra función **set**

1.9 void ReachMatrix::set (char *pi, unsigned j)

Método: set

Clase: Reach Matrix

Tipo: void

Parámetros: Recibe un apuntador a carácter **pi** y un entero positivo **j**.

Llamadas: no hay

Reserva memoria

1.10 bool ReachMatrix::get (unsigned i, unsigned j) const

Método: get

Clase: Reach Matrix

Tipo: booleano

Parámetros: Recibe **i** y **j**.

Llamadas: no hay

Para sacar una dato de la estructura matriz, primero resuelve un apuntador a través de la función **line**.

1.11 bool ReachMatrix::get (char *pi, unsigned j) const

Método: get

Clase: Reach Matrix

Tipo: booleano

Parámetros: Recibe el apuntador a caracteres **pi** y el entero positivo **j**.

Llamadas: no hay

Recibe un apuntador a caracter **pi**, y un **id** de nodo **j** para reservar memoria

1.12 bool ReachMatrix::Add (unsigned n,const ReachSet & from_which,
const ReachSet & which)

Método: Add

Clase: Reach Matrix

Tipo: booleano

Parámetros: un nodo **n**, conjunto de nodos entrantes a **n** llamado **from_which** y un conjunto de nodos salientes de **n**, llamado **which**.

Llamadas: CanAdd, DoAdd

Recibe el nodo que queremos agregar como primitiva semántica **n**, a los conjuntos de nodos entrantes y salientes, y evalúa si es factible agregar dicho nodo llamando a la función **CanAdd** y si es así ejecuta **DoAdd** y regresa el valor de **CanAdd**.

```
1.13 bool ReachMatrix::CanAdd (unsigned n, const ReachSet &
    from_which, const ReachSet & which)
```

Método: CanAdd

Clase: Reach Matrix

Tipo: booleano

Parámetros: un nodo **n**, conjunto de nodos entrantes a **n** llamado **from_which** y un conjunto de nodos salientes de **n**, llamado **which**.

Llamadas: no hay

Recibe un nodo **n**, y los conjuntos de nodos entrantes y salientes representados por **from_which** y **which**. Se buscan los casos en que no es posible considerar primitiva al nodo y en los demás casos regresa el valor de verdad **true**.

Verifica que pueda agregarse un nodo como primitiva semántica. Regresa False en los casos que se listan a continuación:

- Si la palabra se define con la misma palabra, es decir, si entre los nodos de flechas salientes se encuentra el mismo nodo el nodo **n** no puede agregarse como primitiva.

- Si ninguno de los nodos de flechas entrantes o ninguno de los nodos de flechas salientes al nodo en cuestión, es considerado como primitiva, esto es, si todos los nodos apuntados por flechas desde **n** son No-Primitivas, el nodo **n** tampoco lo es.

1.14 void ReachMatrix::DoAdd (unsigned n, const ReachSet & from_which, const ReachSet & which)

Método: DoAdd

Clase: Reach Matrix

Tipo: void

Parámetros: un nodo **n**, conjunto de nodos entrantes a **n** llamado **from_which** y un conjunto de nodos salientes de **n**, llamado **which**.

Llamadas: no hay

Realiza la acción de agregar primitiva

1.15 void ReachMatrix::PrintPrimitives (ostream & f) const

Método: PrintPrimitives

Clase: Reach Matrix

Tipo: void

Parámetros: un archivo de salida **f**

Llamadas: no hay

Recibe un archivo en el que escribirá el identificador de nodo y si el nodo no esta considerado como primitivo, es decir, si la función **IsNonPrimitive** que a su vez verifica que el nodo **i** esté en la cadena **non_primitives**, es falsa, escribe el identificador del nodo en dicho archivo, que es "**Primit.out**"

Escribe los identificadores de las palabras primitivas en el archivo
"primit.out"

1.16 bool IsNonPrimitive (int i)

Método: IsNonPrimitive

Clase: Reach Matrix

Tipo: booleano

Parámetros: entero **i**

Llamadas: no hay

Función booleana que regresa **True** si encuentra al nodo **i** en el conjunto denominado **non_primitives**.

1.17 void RenumberFromTo (const TNetStruct & net, TArray<unsigned>
*from, TArray<unsigned> *to)

Método: RenumberFromTo

Clase: ninguna

Tipo: void

Parámetros: la estructura de red net, el arreglo from y el arreglo to.

Llamadas: no hay

Si un nodo de **net** no está marcado, éste método le asigna los **ItemsIncontainer** del arreglo **to** al apuntador **from** e incremento contador **unmarked**, de otra forma le agrego la constante **UINT_MAX** a dicho apuntador e incrementa el contador **marked**.

1.18 void Renumber (int *dst, const int *src, const TArray<unsigned> & from, unsigned n, int *pn)

Método: Renumber.

Clase: ninguna.

Tipo: booleano

Parámetros: dos apuntadores a entero, **dst** (destino) y **src** (origen), un arreglo **from**, un entero positivo **n** y un apuntador a entero **pn**.

Llamadas: no hay

Reenumera restando 1 a cada número del arreglo **src** y copiándolo al arreglo **dst**.

1.19 void CopyNet (const TNetStruct & net, TNetStruct * net1, const TArray<unsigned> & from, const TArray<unsigned> & to)

Método: CopyNet.

Clase: ninguna.

Tipo: void.

Parámetros: una red **net**, otra red **net1**, un arreglo **from** y un arreglo **to**.

Llamadas: no

Copia cada nodo de **net** especificado en el arreglo **to** a **net1**.

1.20 void sort (const TNetStruct & net, freq_id *freq)

Método: sort.

Clase: ninguna.

Tipo: void

Parámetros: Una red **net** y una estructura **freq_id**

Llamadas: no

Ordena por la frecuencia

1.21 void read_order (int p [], unsigned N)

Método: read_order

Clase: ninguna.

Tipo: void.

Parámetros: recibe un arreglo de entros **p** y un número positivo **N**.

Llamadas: no

Lee el orden de preferencia del archivo **primit.prf**

1.22 void prefsort (const TNetStruct & net, freq_id *freq)

Método: prefsort

Clase: ninguna.

Tipo: void.

Parámetros: una red **net**, una estructura **freq_id**.

Llamadas: no

Para el método 4, calcula la máxima frecuencia de flechas entrantes a un nodo y asigna una frecuencia para cada nodo en función de la frecuencia calculada y la frecuencia máxima

1.23 void AddToPrimitives (ReachMatrix *m, const TNetStruct & net, unsigned i)

Método: AddToPrimitives

Clase: ninguna.

Tipo: void

Parámetros: Una matriz **m**, una red **net**, un entero positivo **i**

Llamadas: no

Recibe la matriz **m**, la estructura net y el número de nodo **i**. Examina el nodo **i** como posible candidato a ser una primitiva semántica. Para ello mantiene dos conjuntos, uno tiene los identificadores de los nodos que tienen flechas entrantes al nodo en cuestión (**from_which**) y el otro tiene los de los nodos que tienen flechas salientes de dicho nodo (**which**). Después llama a un método **Add** de la matriz **m** con tres parámetros: el nodo, el conjunto de nodos con flechas entrantes y el conjunto de nodos con flechas salientes.

```
1.24 void (*sort_fun) (const TNetStruct &, freq_id *);
```

Método: *sort_fun.

Clase: ninguna.

Tipo: void.

Parámetros: una constante TnetStruct y un número de freq_id.

Llamadas: no

Ejecuta alguna de las funciones: **sort** o **prefsort** dependiendo del valor de la variable **sort_fun**.

```
1.25 void OrderPrimitives (ReachMatrix *m, const TNetStruct & net, unsigned  
N)
```

Método: OrderPrimitives

Clase: ninguna.

Tipo: void.

Parámetros: una matriz **m**, una red **net**, un entero positivo **N**.

Llamadas: no

Ordena conforme al identificador de la palabra

1.26 void RandPrimitives (ReachMatrix *m,const TNetStruct & net, unsigned N)

Método: RandPrimitives.

Clase: ninguna.

Tipo: void.

Parámetros: una matriz **m**, una constante **net**, un número positivo **N**.

Llamadas: no

METODO 1: aleatorio, uniforme

Parámetros que recibe: la matriz **m**, la red **net** y el número total de nodos de la red, **N**.

Genera una serie de números aleatorios dentro del rango que caracteriza al tamaño de la red. Después llama a **AddToPrimitives**

1.27 void StatPrimitives (ReachMatrix *m, const TNetStruct & net, unsigned N)

Método: StatPrimitives.

Clase: ninguna.

Tipo: void.

Parámetros: una matriz **m**, una constante **net**, un número positivo **N**.

Llamadas: no

METODO 2: Realiza el análisis en el orden definido **sort**.

Parámetros que recibe: la matriz **m**, la red **net** y el número total de nodos de la red, **N**. Obtiene las frecuencias de cada uno de los nodos en la red dada y los ordena de menor a mayor.

1.28 void RandStatPrimitives (ReachMatrix *m, const TNetStruct & net, unsigned N)

Método: RandStatPrimitives.

Clase: ninguna.

Tipo: void.

Parámetros: una matriz **m**, una constante **net**, un número positivo **N**.

Llamadas: no

METODO 3 : Aleatorio, por frecuencias

Parámetros que recibe: la matriz **m**, la red **net** y el número total de nodos de la red, **N**.

1.29 void PrefStatPrimitives (ReachMatrix *m, const TNetStruct & net, unsigned N)

Método: PrefStatPrimitives

Clase: ninguna.

Tipo: void.

Parámetros: una matriz **m**, una constante **net**, un número positivo **N**.

Llamadas: no

MÉTODO 4: Realiza el análisis en el orden definido por **sort_fun = prefsort**.

Parámetros que recibe: la matriz **m**, la red **net** y el número total de nodos de la red, **N**.

1.30 void ErrorPrimitives (ReachMatrix *, const TNetStruct &, unsigned)

Método: ErrorPrimitives

Clase: ninguna.

Tipo: void.

Parámetros: una matriz **m**, una constante **net**, un número positivo **N**.

Llamadas: no

Si se da una opción distinta al elegir el método, **ErrorPrimitives** envía el mensaje siguiente: "ERROR: No se especificó el método."

1.31 void Primitives (const TNetStruct & net, ostream & f, void (*DoPrimitives) (ReachMatrix *, const NetStruct &, unsigned))

Método: Primitives.

Clase: ninguna.

Tipo: void.

Parámetros: una constante **net**, un archivo de salida **f**, una función determinada por **DoPrimitives**.

Llamadas: no

Parámetros : **net1**, **ostream** (el archivo "**primit.out**"), y **method_name**.

Recibe una estructura de red **net**, un archivo de salida **f** y el tipo de algoritmo o método de procesamiento de la red elegida, al que a su vez le define tres parámetros (la matriz **m**, la red **net** y el número total de nodos de la red, **N**)

Llama a: **DoPrimitives**. Esta es sólo una variable que contiene el número de método elegido. Puede ser: **RandPrimitives**, **RandStatPrimitives**,

StatPrimitives o **PrefStatPrimitives**. También llama a los métodos de ReachMatrix : **Check** y **PrintPrimitives**

Este método crea una matriz **m** de tamaño **N**, identifica las primitivas a través del método elegido (ejecuta el método, pasándole 3 parámetros, la matriz **m**, la red **net** y el número de nodos de la red, **N**) y escribe, tomando como base la misma matriz, las primitivas.

1.32 AnsiString IniFReadString (const char *file, const char *name)

Método: IniFReadString.

Clase: ninguna.

Tipo: AnsiString

Parámetros: un apuntador a caracter file, un apuntador a caracter name.

Llamadas: no

Regresa un entero que indica cuál será el método a ejecutar.

1.33 int IniFReadInteger (const char *file, const char *name)

Método: IniFReadInteger

Clase: ninguna.

Tipo: int

Parámetros: un apuntador a caracter file, un apuntador a caracter name.

Llamadas: IniFReadString

Lee el archivo "**Graph.ini**" que especifica el método a ejecutar.

1.34 void ReadIniFile ()

Método: ReadIniFile
Clase: ninguna.
Tipo: void.
Parámetros: ninguno
Llamadas: IniFReadInteger

Asigna a la variable **algorithm** el número de algoritmo especificado en el archivo "**Graph.ini**".

```
1.35 int main(int argc, char* argv[])
```

Método: main
Clase: ninguna.
Tipo: int
Parámetros: Entre sus argumentos está -c y -v
Llamadas: no

Función main, evalúa los argumentos recibidos.

3.4.5.1 Módulo circle

1. void TNode::BinaryWrite (ofstream &f)

Método: BinaryWrite
Clase: TNode
Tipo: void
Parámetros: archivo de salida **f**
Llamadas: ninguna.

Escribe el identificador de cada nodo, su longitud, si esta marcado o no, el identificador de cada nodo, hacia el que el nodo en cuestión, tiene flechas salientes, y también todos los identificadores de los nodos con flechas entrantes hacia dicho nodo.

2. void TNode::BinaryRead (ifstream &f)

Método: BinaryRead.

Clase: TNode.

Tipo: void

Parámetros: archivo de entrada f

Llamadas: ninguna

Lectura binaria de las características de cada nodo en **f**.

3. void TNetStruct::BinaryWrite (AnsiString name)

Método: BinaryWrite.

Clase: TNetStruct.

Tipo: void.

Parámetros: El nombre de un archivo de salida.

Llamadas: TNode::BinaryWrite

Realiza la escritura de un archivo binario name a través del método **BinaryWrite** de la clase **node**.

4. void TNetStruct::BinaryRead (AnsiString name)

Método: BinaryRead

Clase: TNetStruct

Tipo: void

Parámetros: nombre de archivo **name**.

Llamadas: TNode::BinaryRead

Lectura binaria del archivo **name**, usando el método **BinaryRead** de la clase **node**.

5. void TNetStruct::Visualize (AnsiString fname)

Método: Visualize.

Clase: TNetStruct.

Tipo: void

Parámetros: el nombre del archivo de salida a visualizar.

Llamadas:

Recibe como parámetro el archivo "**net1.vis**" que contiene el diccionario procesado con entradas y salidas, en un formato leíble por el usuario, con las 10358 entradas.

6. void Cleanup (TNetStruct *net, ostream *f)

Método: Cleanup.

Clase: ninguna..

Tipo: void

Parámetros: Una red **net**, un archivo de salida **f**

Llamadas: ninguna.

Depura la estructura de red de los nodos marcados y registra el número de nodos removidos y los que quedan en el archivo **f**.

3.4.5.2 *Módulo vis*

1. void ReadPrimitives (TArray<unsigned> *primitives, istream &f)

Método: ReadPrimitives.

Clase: ninguna.

Tipo: void

Parámetros: el arreglo **primitives**, la entrada **f**.

Llamadas: ninguna.

Lee el archivo **f**, encuentra las primitivas y las agrega al arreglo **primitives**.

2. void MarkPrimitives (const TArray<unsigned> & primitives,const TNetStruct & net1)

Método: MarkPrimitives.

Clase: ninguna.

Tipo: void

Parámetros: el arreglo **primitives**, la red **net1**.

Llamadas: ninguna.

Desmarca todos los nodos de **net1** y vuelve a marcar solo los nodos que son primitivas.

3. void out_word (const char *s1,unsigned n,const TNetStruct & net1,ostream &f,const char *s2)

Método: out_word.

Clase: ninguna.

Tipo: void.

Parámetros: el apuntador a un arreglo **s1**, el número positivo **n**, la red **net1**, el archivo de salida **f** y un apuntador a caracter **s2**.

Llamadas: ninguna.

Imprime la palabra a que se refiere el nodo **n** y su identificador.

```
4. int FindPathLength (unsigned n,const TNetStruct & net,TArray<unsigned> *  
    path,ostream *trace)
```

Método: FindPathLength

Clase: ninguna.

Tipo: int

Llamadas: ninguna.

Tomando como base la red **net**, contruye la trayectoria del ciclo que se forma a partir del nodo **n**, con los identificadores de los nodos del ciclo.

```
5. ret
```

Método: ret

Clase, tipo, parámetros y llamadas: no.

Agrega un elemento individual al arreglo **path**.

```
6. int main(int argc, char* argv[])
```

Método: main

Clase: ninguna.

Tipo: int

Llamadas: ninguna.

Evalúa los argumentos, crea un archivo de salida con las primitivas y los ciclos más pequeños que se relacionan a ellas.

3.4.5.3 Librería *circle*

Clase TNode
Atributos id (n.id), word (n.word), n_of_out (n.n_of_out), dumb (n.dumb), marked (n.marked), tmp_marked (n.tmp_marked), n_pass (n.n_pass), net (n.net), in (0, 0, 100), subst (0, 0, 100), used (0, 0, 100)
Métodos públicos void AddOut (int add_id) void AssignOutToSubst (); void BinaryWrite (ofstream &f); void BinaryRead (ifstream &f); int FindCircles (int id_where, int depth); bool NoCircles (int id_where, int depth); int NumOfReachable (); int NumOfReachableStupid (ofstream &f, bool inverse); int DoNumOfReachable (int depth); void CalcFreqs (bool inverse); bool HasElement (int element); void Substitute (int start_level, int max_level, int level); void IncFreq (int id, int level, int max_level); int n_of_out; int out [max_out]; TArray <TInt> in; TArray <TInt> used; TArray <TInt> subst; bool dumb; bool marked; bool tmp_marked; int n_pass; TNetStruct * net;

Clase TnetStruct¹²

Métodos públicos

```
TNetStruct ()  
~TNetStruct ()  
void EmptyNodes ()  
void FindMaxCircles (ofstream & f);  
void DeleteNonTrans (ofstream & f);  
void AssignOutToSubst ();  
void BinaryWrite (AnsiString name);  
void BinaryRead (AnsiString name);  
void Visualize (AnsiString fname);  
void VisualizeSubst (AnsiString fname, bool JustWords);  
void Substitute (ofstream & f);
```

¹² Para describir las clases se utiliza en este trabajo la notación denominada *Unified Notation*

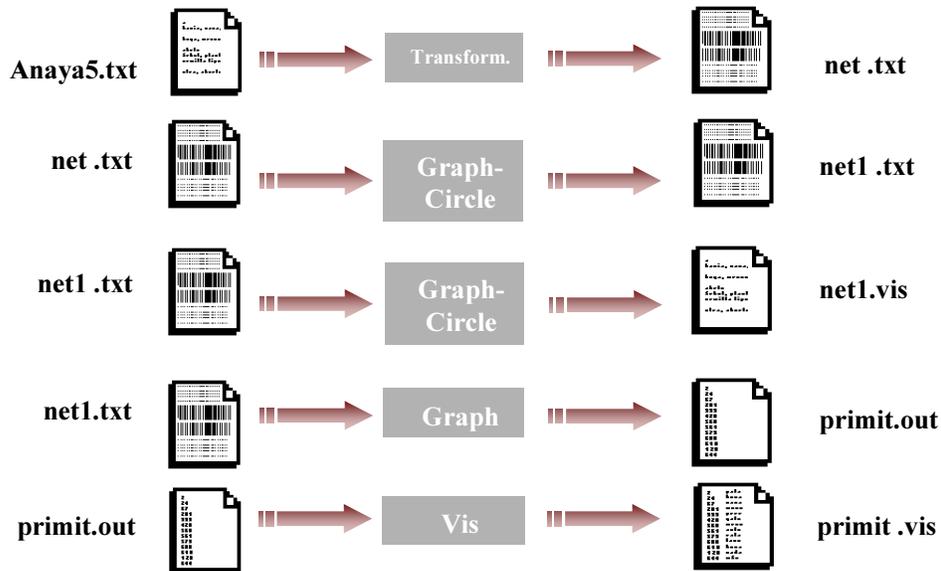


Figura 12. Entradas y salidas del sistema.

Empleando cualquiera de los cuatro métodos (u ordenamientos) establecidos, el sistema toma cada palabra y busca un ciclo entre sus entradas y salidas, pero solo entre las entradas y salidas que ya se encuentren en el grafo sin ciclos (llamaremos a esto "proceso de análisis"). Por esto las primeras palabras tienden a ser no-definidoras, pues al tener menos palabras en el grafo depurado es menos "probable"¹³ la existencia de un ciclo. Conforme el algoritmo va incrementando el número de nodos dentro del grafo sin ciclos, la tendencia a que el proceso de análisis encuentre ciclos es mucho mayor, y las palabras que se analizan al final son las que más fácilmente quedarán en el conjunto mínimo definidor. Por esta razón si el algoritmo toma en primer término el nodo cuya palabra es "abastecer", ésta palabra no tendrá ningún ciclo, se marcará, para distinguirse como parte del grafo sin ciclos. Más tarde le tocará su turno al nodo cuya palabra es "proveer", si

ésta palabra no tiene aún entradas y salidas en el grafo sin ciclos o bien tiene algunas entradas y salidas las cuales no formarían un ciclo con la inclusión de "proveer", ésta palabra se marcaría como parte del grafo sin ciclos y las entradas y salidas en dicho grafo, para éstas palabras, se actualizarían. Posteriormente, si la palabra a analizar es "suministrar", el algoritmo encontraría por lo menos un ciclo al estar evaluando si existe un camino de cada salida de ésta palabra a alguna de sus entradas (hay un ciclo) y no la marcaría, es decir, la consideraría primitiva.

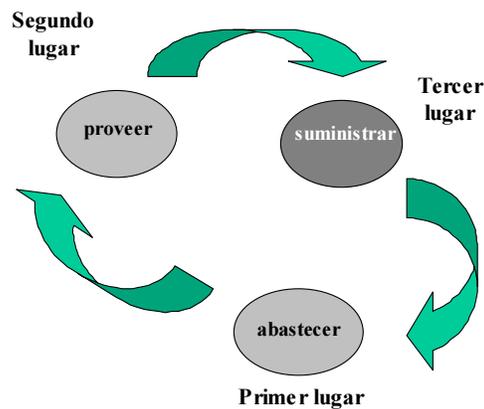


Figura 13. Ejemplo: la palabra seleccionada como primitiva es "suministrar".

En cambio, si la primera palabra que se marca como nodo del grafo depurado, es "suministrar", seguida de "proveer", al llegar el proceso de análisis a "abastecer", el algoritmo encontrará que es "abastecer" la que provoca un ciclo e incluirá ésta como primitiva.

¹³ Sin embargo la probabilidad matemática sólo podría calcularse en función a "temas", "roles temáticos", o a algún elemento de referencia que nos permita medir el grado de relación entre palabras, como se vió en la sección 2.7.

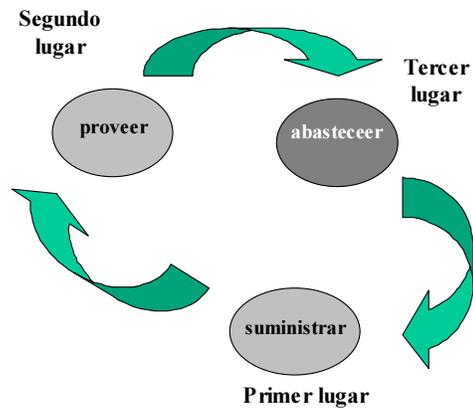


Figura 14. Ejemplo: la palabra seleccionada como primitiva es "abastecer".

De esta manera se ve que, tratándose de las mismas palabras, y las mismas definiciones, el proceso de análisis produce resultados distintos dependiendo de la existencia de salidas de una palabra que conduzcan a entradas a esa misma palabra en el grafo sin ciclos, y a su vez, éste suceso depende del ordenamiento de los nodos a analizar y, por supuesto, de la presencia de los ciclos.

4 Metodología y resultados experimentales

4.1 Metodología experimental

Una vez que el diccionario tradicional ha sido convertido dos tipos distintos de grafos (por lexemas y por significados), hicimos los siguientes experimentos, cuyos resultados se presentan en la siguiente sección y, específicamente, se muestran en la Tabla 2.

Como se explica en la sección 3.3 el resultado del algoritmo depende del ordenamiento de los vértices del grafo. Se deben probar diferentes ordenamientos para obtener el menor conjunto definidor posible, pues no sabemos de ningún algoritmo que encuentre el conjunto de menor tamaño. Probamos los siguientes ordenamientos.

Método 1: aleatorio, uniforme. Usamos el ordenamiento uniformemente aleatorio: en cada iteración del algoritmo A, elegimos aleatoriamente el siguiente vértice $\sigma(i)$ de los vértices todavía no procesados.

Método 2: por frecuencias. Ordenamos los vértices por la frecuencia de su uso en las definiciones del mismo diccionario –es decir, por el número de las flechas entrantes– de menor a mayor. Entonces, los vértices con menor frecuencia

tendieron a entrar en \mathbf{G}' y los de mayor frecuencia tendieron a ser definidores y entrar en \mathbf{P} . Esperábamos que con esta heurística, \mathbf{P} sería menor porque los vértices que lo forman rompen más ciclos en \mathbf{G} .

Método 3: aleatorio, por frecuencias. Este método es una combinación de los métodos 1 y 2. Usamos el ordenamiento aleatorio, pero con las probabilidades en función inversa a las frecuencias: en cada iteración del algoritmo A, elegimos el siguiente vértice $\sigma(i)$ aleatoriamente de los vértices todavía no procesados, siendo la probabilidad del vértice i de ser elegido $p_i \sim f_{max} - f_i + 1$, donde f_i es la frecuencia del vértice como en el método 2, f_{max} es el valor máximo de f_j y \sim significa proporción con normalización, $\sum p_i = 1$. Esperábamos que alguna alternación del ordenamiento rígido del método 2 produciría un conjunto menor.

Método 4: por votación aleatoria. En este método, generamos 20 diferentes conjuntos definidores mínimos \mathbf{P}_i con el método 1, y para cada vértice, contamos el número de los conjuntos \mathbf{P}_i en los cuáles éste entra, asociando así con cada vértice un número entre 20 (entró en todos) y 0 (nunca entró). Ordenamos primero los vértices que nunca entraron en los \mathbf{P}_i usando sus frecuencias, como en el método 2, y después los que entraron, en el orden inverso al número de los \mathbf{P}_i en que entró, desde 1 a 20. Esperábamos que los que entraron en más conjuntos \mathbf{P}_i serían los “mejores” definidores y debieran entrar en el conjunto que buscamos.

Cabe mencionar, que un 80% de los vértices entraron en por lo menos un conjunto \mathbf{P}_i .

Hicimos dos experimentos con dos grafos diferentes.

Grafo 1: por lexemas. En este experimento, consideramos un lexema como un vértice del grafo, es decir, una palabra normalizada morfológicamente. Por ejemplo, las cadenas *piensa*, *pensó*, *pensaríamos* se consideraron equivalentes y se representaron con el vértice *pensar*. No aplicamos ninguna resolución de ambigüedad: así, la cadena *fue* se consideraba como si en lugar de ésta, en el texto

estuviera *ir ser*, es decir, se formaron las flechas desde la palabra encabezado de la definición que contiene *fue* hasta ambos vértices, *ir* y *ser*. Por supuesto, como las entradas ya están en la forma correcta, no se cambiaron. En este método, las definiciones de todos los significados de la palabra –o, en su caso, de todas las entradas homónimas– se consideraron como una larga definición, pues no hay método para distinguir a qué significado u homónimo refiere una palabra en una definición.

Grafo 2: por significados. En este experimento, consideramos un vértice del grafo como un significado específico de la palabra, por ejemplo: $gato_{1a}$ (animales), $gato_{1b}$ (tipo de animales), $gato_{2a}$ (herramienta) etc. Para desambiguar los significados de las palabras que forman las definiciones, empleamos un etiquetador (*tagger*) para la normalización morfológica y para la desambiguación de la parte de oración, y después utilizamos un algoritmo parecido al de Lesk para desambiguar el significado de la palabra en el contexto. Con más detalle este proceso se describe en (Sidorov y Gelbukh, 2001) [42].

En ambos casos, sólo se consideran las palabras significativas, es decir, no se consideran las preposiciones, conjunciones, verbos auxiliares, etc. Nótese que en el Grafo 2, el número de las flechas es exactamente el mismo que el número de las palabras significativas en las definiciones del diccionario, mientras que en el Grafo 1 este número es ligeramente mayor por la ambigüedad léxica.

4.2 Resultados experimentales

Para nuestros experimentos usamos el Diccionario de la lengua española del grupo Anaya (1997)[11]. Este diccionario contiene 30971 entradas, de los cuales sólo

30725 son palabras significativas. Estas palabras significativas se dividen en 60818 significados específicos¹⁴. Estas cifras se muestran en la Tabla 2.

Los resultados de estos experimentos y su discusión se presentan a continuación.

Grafo	En total	No definidores	Depurado
Lexemas	30725	20366	10359
Significados	60818	47802	13016

Tabla 2. Número de vértices en los grafos.

Grafo	Método1 Aleatoriamente, uniforme	Método3 Aleatoriamente, por frecuencias	Método2. Por frecuencias	Método4. Por votación aleatoria
Lexemas	2789, $s = 25$	2770	2302	2246
Significados	2266, $s = 28$	2257	1955	1913

Tabla 3. Número de definidores, con diferentes algoritmos.

Al aplicar el algoritmo de depuración (algoritmo B) a las dos variantes del grafo (por palabras y por significados), encontramos que sólo aproximadamente unos 10 mil vértices están, posiblemente, involucrados en los ciclos (véase Tabla 2).

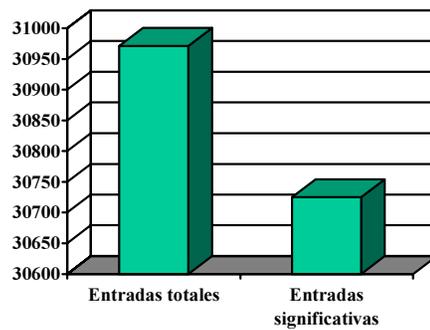
Un resultado inesperado fue que este núcleo de unos 10 mil vértices está muy fuertemente interconectado: prácticamente con cualquier palabra del diccionario, están accesibles, a través de alguna ruta, todas las 10 mil palabras de este núcleo. O sea, al sustituir iterativamente las palabras en la definición con sus propias definiciones, la definición de casi cualquier palabra del diccionario se

¹⁴ En realidad, nuestro algoritmo para la detección de los significados no es perfecto, pero sus errores no son estadísticamente significativos

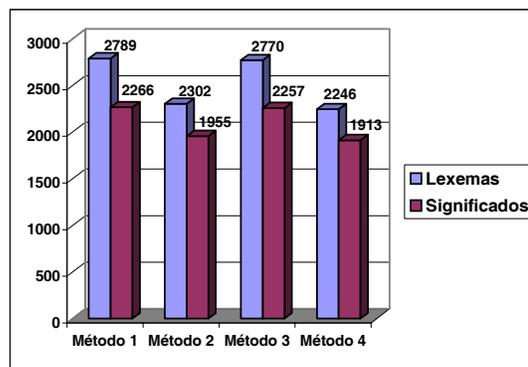
extiende a las mismas 10 mil palabras. Entonces, la tarea de selección del conjunto definidor en un grafo tan fuertemente interconectado es computacionalmente difícil.

A estos dos conjuntos de 10 mil palabras, aplicamos el algoritmo A con las 4 variantes de ordenamiento. Los resultados se presentan en la Tabla 3, mostrándose los tamaños de los conjuntos definidores obtenidos.

Enseguida tenemos podemos observar gráficamente estos resultados:



Gráfica 1. Entradas totales contra entadas significativas



Gráfica 2. Resultados obtenidos por grafo y por método.

Para el método 1, se muestra el promedio de los 20 experimentos y la desviación cuadrática promediada s (s^2 es la dispersión; se sabe que un 67% de los casos se desvían del promedio no más que una s y 99% no más que $3s$). Atrae la atención que la desviación es muy baja, lo que significa que con el método 1, los tamaños de los conjuntos obtenidos son diferentes pero muy parecidos.

Para el método 3, sólo mostramos el resultado de un experimento. Fue sorprendente que estos resultados casi no difieran de los obtenidos con el método 1, a pesar de que las probabilidades en este caso correspondieron a las frecuencias del método 2. Este último mostró un muy buen desempeño produciendo los conjuntos definidores mucho más pequeños. Sin embargo, el método heurístico – método 4– produjo los conjuntos más pequeños que hemos obtenido.

Aunque creemos que con métodos más sofisticados, se puede obtener conjuntos aun más pequeños, no esperamos que el tamaño mínimo del conjunto definidor sea mucho menor que los que hemos obtenido, de aproximadamente 2000 palabras. Esto se debe a las siguientes consideraciones lingüísticas: según la opinión común, se supone que 2 mil es el número de las palabras primitivas, por lo menos en el sentido común, suficientes para definir todas las demás palabras del vocabulario general. Por ejemplo, éste es el número de palabras en el vocabulario definidor de Longman (*Longman defining vocabulary*). También, según nuestro conocimiento, éste es el número de los jeroglíficos en el vocabulario chino básico. Creemos que el hecho de que el tamaño de nuestro conjunto definidor tan exactamente corresponda a la cifra prevista –2 mil– es muy significativo.

Consideremos unos ejemplos de las palabras elegidas con el Método 4, Grafo 1. Aquí están las 20 palabras con la mayor frecuencia de las flechas entrantes (las mejores):

cosa, persona, acción, hacer, efecto, tener, parte, no, conjunto, dar, forma, cierto, cuerpo, relativo, nombre, poder, uno, formar, producir, animal, común, general, determinado, poner, estado, tiempo, decir, planta, obra, etc.

Estas palabras son buenas candidatas a primitivas semánticas. También hay palabras que entraron en el conjunto definidor porque están involucradas en ciclos cortos, aunque tienen una frecuencia baja (las peores):

almuerzo, almuédano, almanaque, alinear, algarroba, alarmar, ahíto, etc.

Se supone que estas palabras son buenos indicadores de la necesidad de cambiar las definiciones en el diccionario para que se excluyan de la lista de las definidoras.

Finalmente, algunas palabras tienen que estar en cualquier conjunto definidor pues tienen lazos. Éstas indican que hay algún problema con el algoritmo de la identificación de las palabras, o que hay una mala definición en el diccionario; en nuestro caso son 47:

ático, borgoña, lapón, etc.

Es interesante investigar la longitud de los ciclos en los cuales estaban involucradas las palabras definidoras si fuesen insertadas en el diccionario, es decir, estos ciclos no contienen las demás palabras del conjunto definidor elegido. Ejemplos de tales ciclos son:

1: *ático* → *ático*

2: *premura* → *prisa* → *premura*

3: *grano* → *cereal* → *centeno* → *grano*

etc. Las longitudes de los ciclos en nuestro ejemplo se distribuyeron como sigue (donde L es la longitud del ciclo más corto y N es el número de las palabras en tales ciclos):

$$\frac{\quad}{L \quad N} \quad \frac{\quad}{L \quad N} \quad \frac{\quad}{L \quad N}$$

1	47	7	53	13	19
2	1496	8	58	14	9
3	177	9	45	15	8
4	67	10	38	16	12
5	47	11	29	17	11
6	72	12	32	18	3

Tabla 4. Longitudes de los ciclos más cortos (ejemplo).

Sólo mostramos aquí los primeros 18 elementos. El ciclo más largo fue de longitud 52.

A continuación tenemos un ejemplo de lo que sucede con algunas palabras que denotan parentesco entre personas. Seleccionamos algunas palabras del grafo por significados, ya depurado (palabras como "hijo", "hermano", "madre", etc., las cuales suponemos -y comprobamos- que son palabras con ciclos cortos). En la Tabla 5, que se encuentra a continuación, tenemos una lista de estas palabras. El método 4, como ya se explicó en la sección 4, consiste en obtener con el método 1 (método aleatorio), 20 conjuntos de primitivas y después determinar el número de veces que cada palabra del grafo aparece en estos conjuntos, ordenando de menor a mayor este criterio, y enseguida tomar como segundo criterio de ordenamiento la frecuencia de las palabras en el grafo (sus flechas entrantes), también de menor a mayor. Esto permite que las palabras que más hayan aparecido en los distintos conjuntos de primitivas obtenidas por el método aleatorio, seguidas de las que tienen una frecuencia alta de uso en las definiciones, son las que tenderán en mayor medida a quedar en el nuevo conjunto de primitivas obtenidas por el método 4. Veamos cuales son las palabras que más aparecen en los conjuntos P obtenidos con el método 1, dato seguido por su frecuencia de uso en las definiciones:

Palabra	Número de veces que aparece en 20 conjuntos P obtenidos por el método 1	Frecuencia
Antecesor	0	1
Cónyuge	0	1
Pariente	0	1
Hermano	0	1
Marido	0	2
Antepasado	0	3
Descendiente	0	5
Novio	1	1
Sobrino	1	1
Tío	2	2
Esposo	3	1
Primo	3	2
Parentela	5	2
Padre	9	4
Parentesco	9	6
Mujer	10	12
Madre	12	1
Familia	12	4
Hijo	13	4

Tabla 5. Algunos ejemplos de datos de palabras relacionadas y sus frecuencias.

Aplicando el método 4, es decir, ordenando el procesamiento de las palabras del diccionario primero por el número de veces que aparecen en 20 conjuntos P obtenidos con el método 1 y luego por la frecuencia de la palabra en las definiciones del diccionario, obtenemos un conjunto de 1913 primitivas entre las que se encuentran:

Parentesco
Padre
Mujer
Madre
Familia
Hijo

Tabla 6. Algunas primitivas halladas por el método 4.

Como se ve en esta lista, las palabras que más veces aparecen en los conjuntos P generados por el método aleatorio son las que tienden a entrar en nuestro conjunto final de primitivas generadas con el método 4.

5 Conclusiones

Presentamos un método y el algoritmo correspondiente, del conjunto mínimo de palabras a través de las cuales se pueden definir todas las demás palabras en un diccionario explicativo; a un conjunto con tal característica lo llamamos un conjunto definidor.

La construcción de tal conjunto se necesita para la conversión del diccionario tradicional en un diccionario semántico computacional (“para la máquina”), siendo un rasgo de éste último, que no se permiten los ciclos en las definiciones.

Nuestro método permitió la construcción de una herramienta que detecta los problemas y defectos en las definiciones del diccionario, relacionados con la presencia de los ciclos, y ayuda al lexicógrafo a corregirlos.

También creemos que el método puede ser útil para la discusión teórica del tema de las primitivas semánticas. Pero las posibles interpretaciones e implicaciones lingüísticas del conjunto **P** obtenido son el tema de investigación futura.

5.1 Resultados obtenidos

Basándonos en el algoritmo arriba descrito, desarrollamos una herramienta que permite al lexicógrafo investigar la estructura del diccionario, proporcionándole la siguiente información:

- Muestra varias características de la palabra, tales como su frecuencia (la cantidad de las flechas entrantes), el tamaño de la definición (la cantidad de las flechas salientes), el largo mínimo del ciclo en que está involucrada, etc.
- Genera diferentes conjuntos definidores mínimos permitiéndole al usuario seleccionar las variantes del algoritmo (las cuales están en función del ordenamiento de las palabras que se analizan).
- Permite al lexicógrafo cambiar manualmente el conjunto generado y verifica que el conjunto cambiado todavía es un conjunto definidor y si éste es mínimo.
- Permite al lexicógrafo cambiar las definiciones de algunas palabras e investigar el impacto en los conjuntos definidores que se generan.
- Dado un conjunto de palabras que el lexicógrafo quiere que sean *no* definidoras, verificar si son compatibles. Si lo son, genera un conjunto definidor (o varios) que no contenga estas palabras. Si no lo son, muestra qué ciclos hay en este conjunto.
- Dado un conjunto de palabras que el lexicógrafo quiere que sean definidoras, genera un conjunto definidor (o varios) que contenga estas palabras. Si este conjunto no puede ser mínimo, sugiere eliminar ciertas palabras de éste.
- Dado un conjunto definidor mínimo, la herramienta puede:

- Para una palabra no definidora, mostrar su definición expandida a las palabras definidoras, es decir, que consiste sólo de las palabras definidoras.
- Para una palabra definidora, mostrar los ciclos (más cortos o todos) que esta palabra causaría si formase parte del diccionario, dada su definición actual.
- La interpretación de los resultados se discute en la sección 4, donde se menciona que en un buen diccionario, el conjunto definidor no debe tener muchas palabras con frecuencia baja, lo que ayuda al lexicógrafo a detectar y corregir ciertos defectos cambiando las definiciones.

5.2 Trabajo futuro

Las tareas principales a investigar se agrupan en dos clases de problemas:

- El problema lingüístico: profundizar en la interpretación lingüística de los resultados y
- El problema técnico: el diseño del algoritmo que encuentre un conjunto P óptimo en un sentido (técnico y/o lingüístico) dado.
- Con más detalle, las tareas futuras específicas son las siguientes:
 - Dar una interpretación lingüística clara al conjunto P obtenido.
 - Elaborar criterios lingüísticos que permitan preferencias en el proceso de inclusión de las palabras en el conjunto definidor (“querer” que una palabra sea o no sea primitiva).
 - Ponderar dichas preferencias (pues no cualquier conjunto de palabras se puede incluir, o no, en un conjunto definidor mínimo) y realizar un algoritmo para la selección del conjunto P en cierto sentido óptimo.

- Desarrollar un algoritmo, exacto o aproximado, para la construcción de un conjunto P del tamaño más pequeño, y analizar su complejidad.

6 Referencias

1. Allport, D. A. (1985). Distributed Memory, Modular Subsystems and Dysphasia, in *Current Perspectives in Disphasia*, Newman, S., Epstein, R. (Eds.). Edinburgh: Churchill Livingstone.
2. Apresjan, J. D. (1974) Regular polysemy, *Linguistics*, 142: 5-32.
3. Apresjan, J. D. (1995) Selected works (in Russian). Moscow, V 1, 472 p., V 2, 768 p.
4. Bar-Hillel, Y. (1964), '*A demonstration of the nonfeasibility of fully automatic high quality machine translation*', in *Language and Information: Selected Essays an Their Theory and Application*. Reading: Addison-Wesley.
5. Budanitsky, Alexander. Lexical semantics relatedness and its application in natural language processing Technical Report CSRG-390. August 1999. Computer System Research Group. University of Toronto <ftp://ftp.cs.utoronto.ca/csr-g-technical-reports/390>
6. Budanitsky, A. (1999) *Lexical semantics relatedness and its application in natural language processing*. Technical report CSRG-390, University of Toronto, Toronto, 146 p.
7. Butterworth, B. (1980). Evidences from Pauses in Speech, *Language Production: Speech and Talk, Vol. I*. London:Academic Press, pp. 155-176.

8. Collins, A. M., Quillian, M.R. (1972). How to Make a Language User, in Organization of Memory, Tulving, Donaldson (Eds.), New York: Academic Press, pp. 309-351.
9. Collins, Allan M, Loftus, Elizabeth F. (1975). A spreading activation theory of semantic processing. *Psychological Review*, 82(6):407-428.
10. Conrad, C., "Cognitive Economy in Semantic Memory", *Journal of Experimental Psychology*, Volume 92, pg 149-154, 1972.
11. Diccionario Anaya de la lengua, Grupo Anaya, Internet, Marzo 1997.
12. Dolan, W., Vanderwende, L., and Richardson, S. (1999) 'Polysemy in a Broad-Coverage Natural Language Processing System', in *Polysemy: Theoretical and Computational Approaches*. (ed.)
13. Evens, M. N. (ed.) (1988) *Relational models of lexicon: Representing knowledge in semantic network*. Cambridge: Cambridge University Press.
14. Fellbaum, C. (1990) The English verb lexicon as a semantic net. *International Journal of Lexicography* 3: 278-301.
15. Fillmore, J. (1982), 'Towards a descriptive framework for spatial deixis', in R.J. Jarvella and W. Klein (eds.), *Speech, Place and Action*. New York: Wiley.
16. Fellbaum, C. (1998) *WordNet, An Electronic Lexical Database*. Cambridge, Mass.: MIT Press, 423 p.
17. Forster, K.I. (1976). Accessing the Mental Lexicon, in *New Approaches to Language Mechanisms*, R.J. Wales and Walker(Eds.), Amsterdam:North Holland, pp. 257-287.
18. Gale, W., Curch, K.W. and Yarowsky, D. (1992), 'A method for disambiguating word senses in a large corpus', *Computers and Humanities*, 26:416-39.
19. Garrett, M.F. (1982) Production of Speech: Observations from Normal and Pathological Language Use, in *Normality and Pathology in Cognitive Functions*, Ellis, A. W. (Ed.), London: Academy Press.

20. Goddard, Cliff. (1999), Polisemy: a problem of definition. In Yael Ravin and Claudia Leacock *Polysemy: Theoretical and computational approaches*. Oxford: Oxford University Press, pp.129-151.
21. Guthrie, J. A., Guthrie, L., Wilks, Y., and Aidinejad, H. (1991), 'Subject-dependent co-occurrence and word sense disambiguation', in *Proceedings of the 29th Annual Meeting of the Association for Computational Linguistics*. Morristown, NJ: ACL, 146-52.
22. Guthrie L., Pustejovsky J., Wilks Y., and Sinator B.M.(1996) The role of lexicons in natural language processing, *Communications of the ACM*, 39(1):63-72, January 1996.
23. Hirst, G. (1987) *Semantic interpretation and resolution of ambiguity*. Cambridge, Cambridge University Press.
24. Katz, J. J. (1972), *Semantic Theory*. New York. Harper & Row.
25. Katz, J. J., Fodor, J.A. (1963) "The structure of a semantic theory", *Language*, 39; 179-210.
26. Katz, J. J., Leacock, C. and Ravin, Y. (1985) 'A decompositional approach to modification', in e. LaPore and B. MacLaughlin(eds.), *Action and Events*. Oxford: Blackwell.
27. Kozima, H. and Furugori, T. (1993) Similarity between words computed by spreading activation on an English dictionary. In: Proceedings of the 6 conference of the European chapter of ACL, pp. 232-239.
28. Kozima, H. and Ito, A. (1997) Context-sensitive word distance by adaptive scaling of a semantic space. In: Mitkov, R. and Nicolov, N. (eds.) Recent Advances in Natural Language Proceedings: Selected Papers from RANLP'95, pp. 111-124.
29. Leacock, G., Chodorow, M.S., and Miller, G.A. (1998). 'Using corpus statistic and WordNet relations for sense identification', *Computational Linguistics*, 24.1:47-65.

30. Lesk, M. (1986), 'Automatic sense disambiguation: how to tell a pine cone form an ice cream cone', in *Proceedings of the 1986 SIGDOC Conference*. New York: Association for Computing Machinery, 24-6.
31. Marslen-Winslow W. (1984). Function and Process in *Spoken Word Recognition: A Tutorial Review*, in Bouma and Bouwhuis (eds.), pp. 125-150.
32. Mel'cuk, I. And Zholkovsky, A. (1988), 'The explanatory combinatorial dictionary' , in M. W. Enens (ed.), *Relational Models of the Lexicon: Representing Knowledge in Semantic Networks*. Cambridge: Cambridge University Press.
33. Miller, G.A., Johnson-Laird, P.N. (1976). *Language and Perception*, Cambridge, Mass.: The Belknap Press of Harvard University Press.
34. Morton, J. (1982). *Disintegrating the Lexicon: An Information Processing Approach*, in Mehler, Walker and Garrett (Eds.)(1982), pp.89-109.
35. Osgood, C.E. (1952). The nature and measurement of meaning. *Psychological Bulletin*, 49:197-237.
36. Pustejovsky, J. (1991). The Generative Lexicon, in *Computational Linguistics*, vol, 17 no. 4, pp. 409-441.
37. Quilliant, M. Ross. (1968) Semantic memory. In M. Minsky, editor, *Semantic Information Processing*. MIT Press, Cambridge, MA.
38. Ravin, Y. and Leacock, C. (2000) *Polysemy: An Overview*. In Yael Ravin and Claudia Leacock *Polysemy: Theoretical and computational approaches*. Oxford: Oxford University Press, 227 p.
39. Resnik, Phillip (1995). Using information content to evaluate semantic similarity. In *Proceedings of the 14th International Conference on Artificial Intelligence*, pages 448-453, Montreal, Canada, August 1995.
40. Rumelhart. D E., Smolensky, J.L., McClelland, J.L., Hinton, G.E. (1986). Schemata and Sequential Thought Processes in PDP models, in *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*

Volume I; Foundations, Rumelhart, D. E., McClelland, J.L. (Eds.), Cambridge, Mass.: MIT Press.

41. Saint-Dizier, P. and Viegas, E. (eds.) (1995) *Computational lexical semantics*. Cambridge: Cambridge University Press, 447 p.
42. Sidorov, Grigori, y Alexander Gelbukh (2001). Desambiguación de los sentidos de las palabras en un diccionario explicativo de español, usando el algoritmo mejorado de Lesk. Sometido al SEPLN-2001, a publicarse.
43. Towell, G., and Voorhees, E.M. (1998). 'Disambiguating highly ambiguous words', *Computational Linguistics*, 24.1:125-46.
44. West M. P. (1953). *A general service list of english words, with semantic frequencies and a supplementary word-list for the writing of popular science and technology*. Longman, Harlow, Sussex.
45. Wierzbicka, A. (1980). *Lingua Mentalis: The semantics of natural language*. New York: Academic Press.
46. Wierzbicka, A. (1990). 'Prototypes save: on the uses and abuses of the notion of "prototypes" in linguistics and related fields', in S. L. Tsohatzidis (ed.), *Meanings and Prototypes: Studies in Linguistic Categorization*. London: Routledge & Kegan Paul.
47. Wierzbicka, A. (1996). *Semantics: Primes and Universals*. Oxford: Oxford University Press.
48. *WordNet: an electronic lexical database*. (1998). C. Fellbaum (ed.), MIT, 423 p.
49. Wittgenstein, L. (1953). *Philosophical Investigations*. Oxford: Basil Blackwell & Mott.

Anexo 1. Código fuente de las funciones principales

6.1 Módulo `graph.cpp`

```
#include "circle.h"

//-----
USEUNIT("circle.cpp");
//-----

class ReachSet
{
public:
    ReachSet () : s (0, 0, 1000) {}
    ~ReachSet () {}

    unsigned GetItemsInContainer () const { return s.GetItemsInContainer
(); }
    void      Flush ()                { s.Flush (); }
    void      Add   (int n)            { s.Add (n); }

    unsigned operator [] (unsigned n) const { return s [n]; }

private:
    TArray<unsigned> s;
};

class ReachMatrix
{
public:
    ReachMatrix (unsigned size);
    ~ReachMatrix ();
};
```

```

    void Flush                ();
    bool Add                  (unsigned n, const ReachSet & from_which, const
ReachSet & which);
    bool Check                () const;
    void PrintPrimitives     (ostream & f) const;
    bool IsNonPrimitive      (int i) const { return get (non_primitives, i);
}

private:

    bool    CanAdd   (unsigned n, const ReachSet & from_which, const
ReachSet & which);
    void    DoAdd    (unsigned n, const ReachSet & from_which, const
ReachSet & which);
    void    or_line  (unsigned i, unsigned j);    // from to
    void    or_line  (char *pi,  char *pj);      // from to
    char    *line    (unsigned i) const;
    void    set      (unsigned i, unsigned j);
    void    set      (char *pi,  unsigned j);
    bool    get      (unsigned i, unsigned j) const;
    bool    get      (char *pi,  unsigned j) const;
    unsigned cols;    // columnas (in ints)
    unsigned lins;    // lineas
    buftype *m;
    char    *non_primitives;
};

ReachMatrix::ReachMatrix (unsigned size)
:
    cols ((size - 1) / (8 * sizeof *m) + 1),
    lins (size),
    m (NULL)
{
    m          = new buftype [cols * lins];
    non_primitives = new char [(lins - 1) / 8 + 1];
    Flush ();
};

bool ReachMatrix::Add (unsigned n,
                      const ReachSet & from_which,
                      const ReachSet & which)
{
    bool ret = CanAdd (n, from_which, which);

    if (ret)
        DoAdd (n, from_which, which);

    return ret;
}

bool ReachMatrix::CanAdd (unsigned n, const ReachSet & from_which, const
ReachSet & which)
{
    for (unsigned i = 0; i < which.GetItemsInContainer (); i++)
    {
        if (which [i] == n)    // example: anochecer is defined through
itself
            return false;
    }
}

```

```

char *l = line (which [i]);

for (unsigned j = 0; j < from_which.GetItemsInContainer (); j++)
{
    assert (from_which [j] != n);

    if (get (l, from_which [j]))
    {
        assert (IsNonPrimitive (which [i]));
        assert (IsNonPrimitive (from_which [j]));

        return false;
    }
    else if (which [i] == from_which [j] && IsNonPrimitive (which
[i]))
    {
        return false;
    }
}

return true;
}

void ReachMatrix::DoAdd (unsigned n,
                        const ReachSet & from_which,
                        const ReachSet & which)
{
    set (non_primitives, n);

    char *l = line (n);

    for (unsigned i = 0; i < which.GetItemsInContainer (); i++)
    {
        unsigned w = which [i];

        if (IsNonPrimitive (w))
        {
            set (l, w); // set w in n
            or_line (w, n); // add w to n

            assert (!get (n, n));
        }
    }

    for (unsigned i = 0; i < from_which.GetItemsInContainer (); i++)
    {
        unsigned fw = from_which [i];

        assert (fw != n);

        if (IsNonPrimitive (fw))
        {
            for (unsigned j = 0; j < lins; j++)
            {
                if (get (line (j), fw) || j == fw)
                {
                    assert (j != n);

                    set (j, n);
                }
            }
        }
    }
}

```

```

        or_line (l, line (j));

        if (get (j, j))
            cerr << "ERROR #2 at j=" << j << endl;

        assert (!get (j, j));
    }
}

}

}

void ReachMatrix::PrintPrimitives (ostream & f) const
{
    for (unsigned i = 0; i < lins; i++)
        if (!IsNonPrimitive (i))
            f << i << endl;
}

void CopyNet (const TNetStruct & net,
              TNetStruct * net1,
              const TArray<unsigned> & from,
              const TArray<unsigned> & to)
{
    int N = to.GetItemsInContainer ();

    cerr << "Copy " << N << " nodes" << endl;

    for (int i = 0; i < N; i++)
    {
        TNode & node = *(TNode*) net.Nodes->Objects [to [i]];

        TNode *node1 = new TNode (node);

        Renumber (& node1->in,    node.in,    from);
        Renumber (& node1->used,  node.used,  from);
        Renumber (& node1->subst, node.subst, from);
        Renumber (node1->out,    node.out,    from, node.n_of_out, &
node1->n_of_out);

        node1->net = net1;
        node1->id = i;

        net1->Nodes->AddObject ("", (TObject *) node1);
    }
}

struct freq_id
{
    unsigned freq;
    unsigned id;
};

void sort (const TNetStruct & net, freq_id *freq)
{
    unsigned N = net.Nodes->Count;

    for (unsigned i = 0; i < N; i++)
    {
        TNode & node = *(TNode*) net.Nodes->Objects [i];

```

```

        freq [i].freq = node.in.GetItemsInContainer () + node.n_of_out;
        freq [i].id   = i;
    }

    qsort (freq, N, sizeof*freq, fcmp);
}

void read_order (int p [], unsigned N)
{
    const char *file = "primit.prf";

    ifstream f (file);

    if (!f.good ())
    {
        cerr << "Cannot read " << file;
        exit (1);
    }

    for (unsigned i = 0; i < N; i++)
        p [i] = -1;

    while (!f.eof ())
    {
        char buf [1000];

        f.getline (buf, sizeof buf);

        if (*buf == '\\0')
            continue;

        int freq = -1, id = -1;

        if (sscanf (buf, "%d %d", &freq, &id) != 2 || freq < 0 || id <
0)
        {
            cerr << "Error #1 reading " << file;
            exit (1);
        }

        if ((unsigned) id >= N || p [id] >= 0)
        {
            cerr << "Error #2 reading " << file;
            exit (1);
        }

        p [id] = freq;
    }
}

void prefsort (const TNetStruct & net, freq_id *freq)
{
    unsigned N = net.Nodes->Count;

    int *order = new int [N];

    read_order (order, N);

    unsigned max_freq = 0;

```

```

for (unsigned i = 0; i < N; i++)
{
    TNode & node = *(TNode*) net.Nodes->Objects [i];

    unsigned fr = node.in.GetItemsInContainer (); // + node.n_of_out;

    if (fr > max_freq) // + node.n_of_out;
        max_freq = fr;
}

if (max_freq == 0)
{
    cerr << "Error # 3" << endl;
    exit (1);
}

for (unsigned i = 0; i < N; i++)
{
    TNode & node = *(TNode*) net.Nodes->Objects [i];

    unsigned fr = node.in.GetItemsInContainer (); // + node.n_of_out;

    freq [i].freq = order [i] >= 0 ? max_freq + order [i] : fr;
    freq [i].id = i;
}

qsort (freq, N, sizeof*freq, fcmp);

delete [] order;
}

void AddToPrimitives (ReachMatrix *m,
                    const TNetStruct & net,
                    unsigned i)
{
    TNode & node = *(TNode*) net.Nodes->Objects [i];

    ReachSet from_which, which;

    for (unsigned j = 0; j < node.in.GetItemsInContainer (); j++)
        from_which.Add (node.in [j]);

    for (int j = 0; j < node.n_of_out; j++)
        which.Add (node.out [j]);

    m->Add (i, from_which, which);
}

void (*sort_fun) (const TNetStruct &, freq_id *);

void OrderPrimitives (ReachMatrix *m,
                    const TNetStruct & net,
                    unsigned N)
{
    freq_id *freq = new freq_id [N];

    sort_fun (net, freq);

    for (unsigned j = 0; j < N; j++)
    {
        if ((j % 100) == 0)

```

```

        cerr << j << " of " << N << "\r";

        int i = freq [j].id;
        AddToPrimitives (m, net, i);
    }

    delete [] freq;
}

void StatPrimitives (ReachMatrix *m,
                    const TNetStruct & net,
                    unsigned N)
{
    cout << "StatPrimitives" << endl;

    sort_fun = sort;

    OrderPrimitives (m, net, N);
}

void PrefStatPrimitives (ReachMatrix *m,
                        const TNetStruct & net,
                        unsigned N)
{
    cout << "PrefStatPrimitives" << endl;

    sort_fun = prefsort;

    OrderPrimitives (m, net, N);
}

void RandStatPrimitives (ReachMatrix *m,
                        const TNetStruct & net,
                        unsigned N)
{
    cout << "RandStatPrimitives" << endl;

    freq_id *freq = new freq_id [N];

    sort (net, freq);

    unsigned max_freq = freq [N-1].freq;          // indexación

    for (unsigned j = 0; j < N; j++)
        freq [j].freq = max_freq - freq [j].freq + 1;

    int total = 0;

    for (unsigned j = 0; j < N; j++)
    {
        total += freq [j].freq;

        if (total < 0)
        {
            cerr << "ERROR: OVERFLOW" << endl;
            exit (1);
        }
    }

    randomize ();
}

```

```

for (unsigned j = 0; j < N; j++)
{
    if ((j % 100) == 0)
        cerr << j << " of " << N << "\r";

    // choice of i

    int k = random (total);

    unsigned i;

    for (i = 0; i < N; i++)
    {
        k -= freq [i].freq;

        if (k < 0)
            break;
    }

    assert (i < N);

    total -= freq [i].freq;
    freq [i].freq = 0;

    AddToPrimitives (m, net, i);
}

delete [] freq;
}

void RandPrimitives (ReachMatrix *m,
                    const TNetStruct & net,
                    unsigned N)
{
    cout << "RandPrimitives" << endl;

    randomize ();

    TArray<bool> done (0, 0, 1000);

    for (unsigned j = 0; j < N; j++)
        done.Add (false);

    for (unsigned j = 0; j < N; j++)
    {
        if ((j % 100) == 0)
            cerr << j << " of " << N << "\r";

        // choice of i

        int k = random (N - j);

        unsigned i;

        for (i = 0; i < N; i++)
        {
            if (!done [i])
                k--;

            if (k < 0)

```

```

        break;
    }

    assert (i < N);

    done [i] = true;

    AddToPrimitives (m, net, i);
}

for (unsigned j = 0; j < N; j++)
    assert (done [j] == true);
}

void Primitives (const TNetStruct & net,
                ostream & f,
                void (*DoPrimitives) (ReachMatrix *,
                                     const TNetStruct &,
                                     unsigned))
{
    unsigned N = net.Nodes->Count;

    ReachMatrix m (N);

    DoPrimitives (& m, net, N);

    cerr << "Check matrix..." << endl;

    assert (m.Check ());

    cerr << "Write Primitives... " << endl;

    m.PrintPrimitives (f);
}

int main(int argc, char* argv[])
{
    cout << "Link " __DATE__ " " __TIME__ << endl;

    ReadIniFile ();

    cerr << "Start..." << endl;

    TNetStruct net1;

    if (argc == 2 && !strcmp (argv [1], "c"))
    {
        TNetStruct net;

        cerr << "Reading old network... " << endl;

        net.BinaryRead ("net.txt");

        cerr << "Cleanup... " << endl;

        for (int i = 0; i < net.Nodes->Count; i++)
            ((TNode*) net.Nodes->Objects [i])->marked = false;
    }
}

```

```

Cleanup (& net, NULL);

TArray<unsigned> from (0, 0, 1000);
TArray<unsigned> to   (0, 0, 1000);
cerr << "Copy... " << endl;

RenumberFromTo (net, & from, & to);

CopyNet (net, & net1, from, to);

cerr << "Write " << net1.Nodes->Count << " nodes..." << endl;

net1.BinaryWrite ("net1.txt");

cerr << "Done: copy network only." << endl;

exit (0);
}
else
{
    cerr << "Reading new network... " << endl;

    net1.BinaryRead ("net1.txt");

    cerr << "Read " << net1.Nodes->Count << " nodes." << endl;
}

if (argc == 2 && !strcmp (argv [1], "v"))
{
    cerr << "Visualize..." << endl;

    net1.Visualize ("net1.vis");

    cerr << "Done: visualize only." << endl;

    exit (0);
}

cerr << "Primitives... " << endl;

Primitives (net1, ofstream ("primit.out"),
    Algorithm == 1 ? RandPrimitives :
    Algorithm == 2 ? StatPrimitives :
    Algorithm == 3 ? RandStatPrimitives :
    Algorithm == 4 ? PrefStatPrimitives :
    ErrorPrimitives);

cerr << "Done. " << endl;

return 0;
}

```

6.2 Módulo `circle.cpp`

```
#include "circle.h"
```

```

void TNode::BinaryWrite (ofstream &f)
{
    f.write ((const char *) &id, sizeof(int));

    int len = word.Length();

    f.write ((const char *) &len, sizeof(int));

    for (int i = 1; i <= word.Length(); i++)
        f.write ((const char *) &word [i], sizeof(char));

    f.write ((char *) &marked, sizeof(char));

    f.write ((const char *) &n_of_out, sizeof(int));

    for (int i = 0; i < n_of_out; i++)
    {
        f.write ((const char *) & out[i], sizeof(int));
    }

    int q = in.GetItemsInContainer ();

    f.write ((const char *) &q, sizeof(int));

    for (int i = 0; i < q; i++)
    {
        int q = in[i].i;

        f.write ((const char *) & q, sizeof(q));
    }

    int q1 = used.GetItemsInContainer ();

    f.write ((const char *) &q1, sizeof(int));

    for (int i = 0; i < q1; i++)
    {
        int q = used[i].i;

        f.write ((const char *) & q, sizeof(q));
    }

    int q2 = subst.GetItemsInContainer ();

    f.write ((const char *) &q2, sizeof(int));

    for (int i = 0; i < q2; i++)
    {
        int q = subst[i].i;

        f.write ((const char *) & q, sizeof(q));
    }
}

//-----

void TNode::BinaryRead (ifstream &f)
{

```

```

f.read ((char *) &id, sizeof(int));

int len;

f.read ((char *) &len, sizeof(int));

char buf [65535];

if (len >= 65535 - 1)
{
    assert (len < 65535 - 1);
}

int i;

for (i = 1 - 1; i <= len - 1; i++)
    f.read ((char *) &buf [i], sizeof(char));

buf [i] = '\\0';

word = buf;

f.read ((char *) &marked, sizeof(char));

f.read ((char *) &n_of_out, sizeof(int));

for (int i = 0; i < n_of_out; i++)
    f.read ((char *) & out[i], sizeof(int));

int q;

f.read ((char *) &q, sizeof(int));

for (int i = 0; i < q; i++)
{
    int j;

    f.read ((char *) & j, sizeof(int));

    TInt int_obj = TInt (j);

    in.Add (int_obj);
}

f.read ((char *) &q, sizeof(int));

for (int i = 0; i < q; i++)
{
    int j;

    f.read ((char *) & j, sizeof(int));

    TInt int_obj = TInt (j);

    used.Add (int_obj);
}

f.read ((char *) &q, sizeof(int));

for (int i = 0; i < q; i++)
{

```

```

        int j;

        f.read ((char *) & j, sizeof(int));

        TInt int_obj = TInt (j);

        subst.Add (int_obj);
    }
}

//-----

void TNetStruct::BinaryWrite (AnsiString name)
{
    ofstream f (name.c_str(), ios::binary | ios::trunc);

    int total = Nodes->Count;

    f.write ((const char *) &total, sizeof(int));

    for (int i = 0; i < Nodes->Count; i++)
    {
        TNode * node = (TNode*)Nodes->Objects [i];

        node->BinaryWrite (f);
    }

    f.close ();
}

//-----

void TNetStruct::BinaryRead (AnsiString name)
{
    ifstream f (name.c_str(), ios::binary);

    int total;

    f.read ((char *) &total, sizeof(int));

    int id = 1;

    for (int i = 0; i < total; i++)
    {
        TNode * node = new TNode (this);

        node->BinaryRead (f);

        for (int q = id; q < node->id; q++) // inserción de nodos vacíos
        {
            TNode * node2 = new TNode (q, "", this);

            Nodes->AddObject (q, (TObject*)node2);
        }

        Nodes->AddObject (IntToStr(node->id), (TObject*)node);

        id = node->id + 1;
    }

    f.close ();
}

```

```

}

//-----

void TNetStruct::Visualize (AnsiString fname)
{
    ofstream f (fname.c_str());

    TNode * node0 = (TNode*) Nodes->Objects [0];

    for (int i = 0; i < Nodes->Count; i++)
    {
        TNode * node = (TNode*) Nodes->Objects [i];

        if (node == NULL || node->word == "")
            continue;

        f << node->id << ":" << node->word.c_str() << " (" << node-
>marked << ")" << endl;

        for (int j = 0; j < node->n_of_out; j++)
        {
            TNode * node2 = (TNode*) node->net->Nodes->Objects [node-
>out[j] - node0->id];

            f << node2->id << ":" << node2->word.c_str() << ", ";
        }

        f << endl << endl;

        for (unsigned j = 0; j < node->in.GetItemsInContainer (); j++)
        {
            TNode * node2 = (TNode*) node->net->Nodes->Objects [node-
>in[j].i - node0->id];

            f << node2->id << ":" << node2->word.c_str() << ", ";
        }

        f << endl << endl << "-----" << endl;
    }

    f.close ();
}

//-----

void Cleanup (TNetStruct *net, ostream *f)
{
    if (f != NULL)
        (*f) << "Cleanup(): start." << endl;

    bool changed;
    int removed = 0;

    do
    {
        changed = false;

        for (int j = 0; j < net->Nodes->Count; j++)

```

```

    {
        TNode *node = (TNode*) net->Nodes->Objects [j];

        if (node->marked)
            continue;

        bool has_in = false;
        bool has_out = false;

        for (int i = 0; i < node->n_of_out; i++)
        {
            TNode * n = (TNode*) net->Nodes->Objects [node->out[i] -
1];

            if (!n->marked)
            {
                has_out = true;
                break;
            }
        }

        for (unsigned i = 0; i < node->in.GetItemsInContainer(); i++)
        {
            TNode * n = (TNode*) net->Nodes->Objects [node->in [i].i
- 1];

            if (!n->marked)
            {
                has_in = true;
                break;
            }
        }

        if (!has_in || !has_out)
        {
            node->marked = true;
            changed = true;
            removed++;
        }
    }
}
while (changed);

if (f != NULL)
    (*f) << "Cleanup(): done. Removed " << removed
    << ", remain " << (net->Nodes->Count - removed) << "." <<
endl;
}

//-----

```

6.3 Módulo vis.cpp

```

#include <condefs.h>
#include <iomanip.h>

```

```

#include <classlib/sets.h>
#pragma hdrstop

#include "circle.h"

//-----
USEUNIT("circle.cpp");
//-----

void ReadPrimitives (TArray<unsigned> *primitives, istream &f)
{
    unsigned n;

    while (n = UINT_MAX, f >> n, n != UINT_MAX)
        primitives->Add (n);
}

void MarkPrimitives (const TArray<unsigned> & primitives,
                    const TNetStruct & net1)
{
    unsigned N = net1.Nodes->Count;

    for (unsigned i = 0; i < N; i++)
    {
        TNode & node = *(TNode*) net1.Nodes->Objects [i];

        node.marked = false;
    }

    for (unsigned i = 0; i < primitives.GetItemsInContainer (); i++)
    {
        TNode & node = *(TNode*) net1.Nodes->Objects [primitives [i]];

        node.marked = true;
    }
}

void out_word (const char *s1,
              unsigned n,
              const TNetStruct & net1,
              ostream &f,
              const char *s2)
{
    TNode & node = *(TNode*) net1.Nodes->Objects [n];

    f << s1 << node.word.c_str () << "<" << n << ">" << s2;
}

void PrintStack (TArray<unsigned> stack,
                int depth,
                const TNetStruct & net1,
                ostream &f)
{
    f << "      ";

    for (int i = 1; i < depth; i++) // do not print the headword
    {
        out_word ("", i, net1, f,
                 i == depth - 1 ? "" : " -> ");
    }
}

```

```

    f << endl;
}

void FindCircles (unsigned cur,
                 unsigned src,
                 int depth,
                 TArray<unsigned> & stack,
                 const TNetStruct & net1,
                 ostream &f)
{
    stack [depth] = cur;

    TNode & current = *(TNode*) net1.Nodes->Objects [cur];

    for (int i = 0; i < current.n_of_out; i++)
    {
        unsigned out = current.out[i];

        /*
        char *s_out = ((TNode*) net1.Nodes->Objects [out])->word.c_str
();
        char *s_cur = ((TNode*) net1.Nodes->Objects [cur])->word.c_str
();
        char *s_src = ((TNode*) net1.Nodes->Objects [src])->word.c_str
();
        */

        if (out == src) // found
        {
            PrintStack (stack, depth, net1, f);
        }
        else
        {
            TNode & node = *(TNode*) net1.Nodes->Objects [out];

            if (!node.marked)
            {
                node.marked = true;

                FindCircles (out, src, depth + 1, stack, net1, f);

                node.marked = false;
            }
        }
    }
}

void PrintPaths (unsigned n, const TNetStruct & net1, ostream &f)
{
    out_word ("", n, net1, f, ":\n");

    TArray<unsigned> stack (0, 0, 1000);

    FindCircles (n, n, 0, stack, net1, f);
}

int FindPathLength (unsigned n,
                   const TNetStruct & net,
                   TArray<unsigned> * path,
                   ostream *trace)
{

```

```

TArray<unsigned> buf0 (0, 0, 10000);
TArray<unsigned> buf1 (0, 0, 10000);
TArray<unsigned> *buf [2] = {& buf0, & buf1};

int select = 0;

const unsigned none = UINT_MAX;

unsigned N = net.Nodes->Count;

for (unsigned i = 0; i < N; i++)
{
    buf0.Add (none);
    buf1.Add (none);
}

TNode & current = *(TNode*) net.Nodes->Objects [n];

int iter = 1;

for (int i = 0; i < current.n_of_out; i++)
{
    unsigned out = current.out[i];

    if (out == n)
    {
        (*buf [!select]) [out] = n;
        goto ret;
    }

    TNode & node = *(TNode*) net.Nodes->Objects [out];

    if (!node.marked)
        buf0 [out] = n;
}

if (trace != NULL)
    out_word ("***** word ***** ", n, net, *trace,
"\n");

for (select = 0; ; select = !select)
{
    iter++;

    if (trace != NULL)
        *trace << "----- iter ----- " << iter <<
endl;

    bool changed = false;

    TArray<unsigned> & buf_old = * buf [ select];
    TArray<unsigned> & buf_new = * buf [!select];

    for (unsigned i = 0; i < N; i++)
        buf_new [i] = buf_old [i];

    for (unsigned i = 0; i < N; i++)
    {
        if (buf_old [i] != none)
        {
            TNode & cur = *(TNode*) net.Nodes->Objects [i];

```

```

for (int j = 0; j < cur.n_of_out; j++)
{
    unsigned out = cur.out[j];

    if (out == n)
    {
        buf_new [out] = i;
        goto ret;
    }

    TNode & node = *(TNode*) net.Nodes->Objects [out];

    if (node.marked)
    {
        if (trace != NULL)
        {
            out_word ("* ", i, net, *trace, " -> ");
            out_word ("", out, net, *trace, "\n");
        }
    }

    if (buf_new [out] != none)
    {
        if (trace != NULL)
        {
            out_word (". ", i, net, *trace, " -> ");
            out_word ("", out, net, *trace, "\n");
        }
    }

    if (!node.marked && buf_new [out] == none)
    {
        if (trace != NULL)
        {
            out_word ("> ", i, net, *trace, " -> ");
            out_word ("", out, net, *trace, "\n");
        }

        buf_new [out] = i;
        changed = true;
    }
}
}

if (!changed)
{
    iter = -iter;
    goto ret;
}

ret:

if (path != NULL)
{
    path->Flush ();

    if (iter > 0)
    {

```

```

        unsigned i = n;

        do
        {
            path->Add (i);

            i = (*buf [!select])[i];

            assert (i < N);
        }
        while (i != n);
    }
}

return iter;
}

int main(int argc, char* argv[])
{
    bool verify = false;
    bool list = false;

    if (argc > 1 && !strcmp (argv [1], "-v"))
    {
        argc--;
        argv++;

        verify = true;

        cerr << "Verify mode; no output." << endl;
    }

    if (argc > 1 && !strcmp (argv [1], "-l"))
    {
        argc--;
        argv++;

        list = true;

        cerr << "List mode." << endl;
    }

    istream *in = & cin;
    ostream *out = & cout;

    if (argc > 1)
    {
        in = new ifstream (argv [1]);

        if (!in->good ())
        {
            cerr << "Cannot open input file <" << argv [1] << ">" <<
endl;

            exit (1);
        }
    }

    if (argc > 3)
    {
        cerr << "ERROR: extra parameter." << endl;
    }
}

```

```

        exit (1);
    }

    if (argc > 2)
    {
        if (verify)
        {
            cerr << "ERROR: with -v option, there is no output." << endl;
            exit (1);
        }

        out = new ofstream (argv [2]);

        if (!out->good ())
        {
            cerr << "Cannot open output file <" << argv [2] << ">" <<
endl;
            exit (1);
        }
    }

    cerr << "Start..." << endl;

    TNetStruct net1;

    cerr << "Reading new network... " << endl;

    net1.BinaryRead ("net1.txt");

    cerr << "Read " << net1.Nodes->Count << " nodes." << endl;

    TArray<unsigned> primitives (0, 0, 1000);

    if (!list)
    {
        cerr << "Reading Primitives... " << endl;

        ReadPrimitives (& primitives, *in);
        MarkPrimitives (primitives, net1);
    }

    cerr << "Working... " << endl;

    if (verify)
    {
        unsigned N = net1.Nodes->Count;

        TSet<unsigned> prim (N);

        for (unsigned i = 0; i < primitives.GetItemsInContainer (); i++)
            prim.Add (primitives [i]);

        for (unsigned i = 0; i < N; i++)
        {
            if (i % (N / 200) == 0)
            {
                cerr << (i * 100) / N << "%\r";
            }

            int path = FindPathLength (i, net1, NULL, NULL);

```

```

        if (prim.HasMember (i))
        {
            if (path <= 0)
            {
                cerr << "ERROR: no cycle at " << i << endl;
            }
        }
        else
        {
            if (path > 0)
            {
                cerr << "ERROR: cycle at " << i << endl;
            }
        }
    }
}
else if (list)
{
    unsigned freq, id;

    while (*in >> freq >> id, !in->eof ()) // @@@@ quitar freq
    {
        TNode & node = *(TNode*) net1.Nodes->Objects [id];

        *out << setw (7) << freq << " " // @@@@ node.freq?
            << setw (3) << node.n_of_out
            << setw (5) << node.in.GetItemsInContainer ()
            << " ";

        out_word (" ", id, net1, *out, "\n");
    }
}
else
{
    unsigned Nprim = primitives.GetItemsInContainer ();

    cerr << "Output " << Nprim << " words." << endl;

    TArray<unsigned> path (0, 0, 100);

    for (unsigned i = 0; i < Nprim; i++)
    {
        if (i % (Nprim / 200) == 0)
        {
            cerr << (i * 100) / Nprim << "%\r";
        }

        unsigned n = primitives [i];

        // PrintPaths (n, net1, *out);

        int path_len = FindPathLength (n, net1, & path, NULL);

        assert (path_len == (int) path.GetItemsInContainer ());

        TNode & node = *(TNode*) net1.Nodes->Objects [n];

        *out << setw (4) << path_len << " "
            << setw (3) << node.n_of_out
            << setw (5) << node.in.GetItemsInContainer ()
            << " ";
    }
}

```

```

        out_word (" ", n, net1, *out, " -> ");

        assert (path [0] == n);

        for (int j = path.GetItemsInContainer () - 1; j >= 0; j--)
            out_word ("", path [j], net1, *out, j > 0 ? " -> " : "");

        *out << endl;
    }
}

cerr << "Done.      " << endl;

return 0;
}

```

6.4 Declaraciones: circle.h

```

//-----
#ifndef circleH
#define circleH

// #define STRICT
#include <classlib\arrays>

#include <assert.h>

#include <fstream.h>

const max_out = 100;

//-----

struct rec
{
    int arr [19];
} ;

class TNode
{
public:
    TNode (int a_id, AnsiString a_word, TNetStruct * a_net):
        id (a_id), net (a_net), n_of_out (0), dumb (false),
        marked (false), tmp_marked (false), word (a_word),
        in(20, 0, 40), used(20, 0, 2000), subst(20, 0, 40) {}

    TNode (TNetStruct * a_net):
        id (0), net (a_net), n_of_out (0), dumb (false),
        marked (false), tmp_marked (false), word (""),
        in(20, 0, 40), used(20, 0, 2000), subst(20, 0, 40) {}

    TNode (const TNode &n)
    :
        id          (n.id),

```

```

    word      (n.word),
    n_of_out  (n.n_of_out),
    dumb      (n.dumb),
    marked    (n.marked),
    tmp_marked (n.tmp_marked),
    n_pass    (n.n_pass),
    net       (n.net),
    in        (0, 0, 100),
    subst     (0, 0, 100),
    used      (0, 0, 100)
}

for (int i = 0; i < n_of_out; i++)
    out [i] = n.out [i];

for (unsigned i = 0; i < n.in.GetItemsInContainer (); i++)
    in [i] = n.in [i];

for (unsigned i = 0; i < n.used.GetItemsInContainer (); i++)
    used [i] = n.used [i];

for (unsigned i = 0; i < n.subst.GetItemsInContainer (); i++)
    subst [i] = n.subst [i];
}

~TNode () {}

void AddOut (int add_id)
{
    assert (n_of_out < max_out);

    bool flag = false;

    for (int i = 0; i < n_of_out; i++)
        if (out [i] == add_id
            || add_id == id)
        {
            flag = true;
            break;
        }

    if (!flag)
        out [n_of_out++] = add_id;
}

void BinaryWrite (ofstream &f);
void BinaryRead  (ifstream &f);

int id;

AnsiString word;

int n_of_out;          // Número de nodos de lfechas salientes
int out [max_out];    // Arreglo de salientes (su definición)

TArray <TInt> in;      // Arreglo de entrantes

bool marked;          // Los que no tienen circulos viciosos
bool tmp_marked;

int n_pass;

```

```

    TNetStruct * net;
};

//-----

class TNetStruct
{
public:
    TNetStruct ()
    {
        Nodes = new TStringList ();

        mas = new rec[30800];
    }

    ~TNetStruct ()
    {
        delete mas;

        EmptyNodes ();

        delete Nodes;
    }

    void EmptyNodes ()
    {
        for (int i = 0; i < Nodes->Count; i++)
        {
            TNode * node = (TNode*)Nodes->Objects [i];

            delete node;

            Nodes->Objects [i] = NULL;
        }

        Nodes->Clear ();
    }

    void BinaryWrite (AnsiString name);
    void BinaryRead  (AnsiString name);
    void Visualize   (AnsiString fname);
    void VisualizeSubst (AnsiString fname, bool JustWords);

    void Substitute (ofstream & f);

    TStringList * Nodes;

    rec * mas;
};

//-----

```

Anexo 2. Formato del grafo de entrada

0:ábaco
380:alambre, 860:arquitectura, 1246:base, 1349:bola, 1673:capitel,
2468:contar, 2605:correr, 3495:dórico, 3932:equino, 4630:forma,
4953:griego, 6347:marco, 7126:orden, 7291:paralelepípedo,
7293:paralelo, 9121:servir, 9392:sujeto,

3495:dórico, 3932:equino, 9468:tablero,

1:abajo
1170:bajo, 3353:dirección, 5521:inferior, 6181:lugar,

1169:bajar, 1170:bajo, 1525:caer, 2861:declinar, 4412:falda,
4656:foso, 4797:gancho, 5771:invertir, 6032:levantar, 6390:mate,
7256:palo, 7332:partido, 8210:quebrado,

2:abandonado
3088:descuidado, 5114:higiénico, 7741:poco, 9367:sucio,

3088:descuidado, 6203:madeja,

3:abandonar
171:actividad, 274:afecto, 279:afición, 436:algo, 742:apoyar,
868:arrastrar, 1525:caer, 2253:completo, 2637:cosa, 2905:dejar,
3020:desamparar, 3089:descuidar, 3152:desistir, 3691:empezar,
3878:entregar, 5702:interés, 5789:ir, 6181:lugar, 6448:meditar,
6988:obligación, 7354:pasión, 7553:persona, 8594:renunciar,
9974:uno, 9991:usar, 10201:vicio, 10302:voluntad,

4:abandono, 354:ahorcar, 482:alta, 2151:colgar, 2798:cuña,
3019:desalojar, 3438:disuadir, 3664:emigrante, 3665:emigrar,
4254:evacuar, 4384:extraviar, 7696:plantar, 7934:presionar,

4:abandono
3:abandonar, 98:acción, 2637:cosa, 2904:dejadez, 3017:desaliño,
3565:efecto, 7553:persona,

3663:emigración, 8593:renuncia,

5:abanico
359:aire, 1742:carretera, 1973:ciclismo, 2011:circular,
2600:corredor, 2626:cortar, 3413:disposición, 4536:fila,
4630:forma, 5024:hacer, 5665:instrumento, 5923:lado, 6732:mover,
8894:sable, 9012:sector, 9604:tener, 10212:viento,
4975:guarda, 7253:palmito, 10060:varilla, 10061:varillaje,

6:abaratar
436:algo, 7856:precio, 8335:rebajar,
1169:bajar,

7:abarca
1587:calzado, 2431:consistente, 2594:correa, 2767:cuerda,
8345:reborde, 8884:rústico, 9375:suela, 9391:sujetar,
2174:comarca, 4763:gacela, 5675:integral, 6135:llave,
6445:mediodía, 6772:mundial, 6824:narrativa, 7891:prehistoria,
9046:semana,

8:abarcar
89:acaparar, 404:alcanzar, 1015:atender, 1409:brazo, 1824:caza,
2268:comprender, 2474:contener, 2637:cosa, 8822:rodear,
9311:sorprender, 9604:tener, 9645:terreno, 9670:tiempo,
10265:vista,
2117:coger, 3800:englobar,

9:abastecer
2637:cosa, 6846:necesario, 8116:proveer,
8115:proveedor, 8618:repostar, 9401:suministrar,

10:abatido
623:ánimo, 4726:fuerza, 6900:no, 9604:tener,
732:apocado, 1170:bajo,

11:abatimiento
12:abatir, 98:acción, 3144:desilusión, 3565:efecto, 4569:físico,
6691:moral, 7828:postración, 9902:tristeza,
307:agobiar, 2845:decaimiento,

12:abatir
436:algo, 1116:ave, 1169:bajar, 2637:cosa, 2995:derribar,
3015:desalentar, 3022:desanimar, 3029:desarmar, 3058:descender,
3164:desmontar, 3539:echar, 5024:hacer, 5239:humillar, 5789:ir,
7779:poner, 9601:tendido, 9675:tierra, 10176:vertical,
10291:volar,

11:abatimiento, 723:aplastar, 1055:atropellar, 5239:humillar,
5246:hundir,

13:abdomen

684:aparato, 913:artrópodo, 2474:contener, 2771:cuerpo,
3324:digestivo, 6728:motor, 7321:parte, 7371:pata, 7824:posterior,
8633:reproductor, 9986:urinario, 10213:vientre, 10252:viscera,

14:abdominal, 342:aguijón, 787:arácnido, 1888:cerco,
2718:crustáceo, 4010:escorpión, 4775:galera, 5112:hígado,
5626:insecto, 7063:oído, 9741:tórax, 10252:viscera,

14:abdominal

13:abdomen, 8544:relativo,

790:araña, 5737:intestino, 6367:marsupial,

15:abecedario

1751:cartel, 2393:conjunto, 3695:emplear, 3849:enseñar,
5267:idioma, 5991:leer, 6012:lenguaje, 6024:letra, 6319>manual,
7128:ordenada, 9109:serie, 9146:signo, 9309:sordo,
9569:telegráfico, 9991:usar,

426:alfabeto,

16:abeja

329:agrupar, 1693:caracterizar, 1880:cera, 1983:cierto,
2284:común, 2771:cuerpo, 3443:diverso, 4079:especie, 5459:indicar,
5498:industrial, 5626:insecto, 6561:miel, 6851:néctar,
6914:nombre, 7128:ordenada, 7553:persona, 7758:polen,
8014:producción, 8377:recibir, 9031:segregar, 9228:social,
9604:tener, 9779:trabajador, 10003:utilizar,

1133:avispa, 1172:bala, 1880:cera, 2155:colmena, 6561:miel,
6851:néctar, 7263:panal, 7769:pollo, 8523:reina,

17:abertura

41:abrir, 346:agujero, 359:aire, 585:ancho, 586:anchura,
900:articulación, 2293:conceder, 3283:diámetro, 3565:efecto,
4681:franqueza, 4954:grieta, 5068:hendidura, 6014:lente,
6677:montaña, 7143:órgano, 7359:paso, 9069:sencillez,
9196:sistema, 9858:trato, 9999:útil, 10038:valle,

346:agujero, 1093:automático, 1308:bigote, 1337:boca,
1365:boquilla, 1483:buzón, 1960:chorro, 3104:desembocadura,
3105:desembocar, 3331:dilatación, 4344:explosión, 4954:grieta,
6443:medio, 6743:mucosa, 7017:obturador, 7018:obturar, 7067:ojo,
7151:orificio, 7803:portilla, 7804:portillo, 8196:pupila,
8229:quebra, 8281:ranura, 8509:registro, 8695:resquicio,
9152:silbo, 9433:surco, 9723:toma, 10044:válvula, 10129:ventana,
10130:ventilación, 10131:ventilador, 10133:ventosa, 10294:volcán,
10323:vulva,

18:abeto

128:acicular, 386:alba, 496:alto, 745:apreciar, 799:árbol,
1182:balear, 1325:blanco, 1326:blancura, 2284:común, 2553:copa,
2984:derecho, 3443:diverso, 4079:especie, 4363:extender,
4834:género, 5152:hoja, 5185:horizontal, 6204:madera, 6914:nombre,
7551:persistente, 8273:rama, 8377:recibir, 9501:tamaño,
9912:tronco,

7640:piña,

19:abierto

19:abierto, 41:abrir, 183:acuerdo, 359:aire, 730:aplicar,
900:articulación, 1197:bancario, 1902:cerrado, 1971:cicatrizar,
2044:claro, 2241:competición, 2753:cubierta, 2765:cuenta,
2973:deportivo, 3332:dilatado, 3635:embarcación, 4120:espontáneo,
4268:evolucionar, 4311:exigencia, 4322:expansivo, 4623:fonético,
4679:franco, 5492:indudable, 6131:llano, 6784:muro, 6900:no,
7143:órgano, 7323:participante, 7359:paso, 7374:patente,
7553:persona, 7744:poder, 8064:pronunciar, 8296:raso,
8623:representar, 8712:resultar, 8822:rodear, 8922:saldar,
9179:sincero, 9294:sonido, 9604:tener, 9670:tiempo, 10036:valla,
10287:vocal,

19:abierto, 565:amplio, 825:argolla, 1150:azote, 1339:bocado,
1346:bofetada, 1364:boquiabierto, 1616:canal, 1632:canguro,
4809:gasas, 4946:grave, 5926:ladrón, 6042:liberal, 6051:libre,
6160:logia, 6390:mate, 6443:medio, 7281:parábola, 7714:plaza,
7825:postigo, 8307:ratón, 8851:roseta, 8960:sandalia,
9303:soportal, 9517:tapar, 9928:tubo,

20:ablandar

1327:blando, 1831:ceder, 2398:conmover, 2637:cosa, 3776:enfado,
4701:frío, 4752:furia, 5689:intensidad, 6144:lluvia, 6608:mitigar,
7779:poner, 8784:rigor, 9331:suavizar, 10212:viento,

520:amansar, 8070:propiciar,

21:abnegación

169:actitud, 436:algo, 1288:beneficiar, 2737:cualidad, 2941:demás,
5024:hacer, 7985:privación, 8593:renuncia, 8906:sacrificio,
9304:soportar, 9388:sufrir, 9974:uno, 10303:voluntario,

8906:sacrificio,

22:abogado

2067:cliente, 2881:defender, 3298:dictamen, 3670:emitir,
5702:interés, 5890:jurídico, 5894:justicia, 6429:media,
7553:persona, 7698:plantear, 7997:problema, 8100:protector,
8974:santo, 9890:tribunal,

416:alegar, 1317:birrete, 2806:curia, 2882:defensa, 5547:informar,
5550:informe, 6025:letrado,

23:abolengo

649:antepasado, 922:ascendencia, 4828:general, 5077:herencia,
5291:ilustre, 7390:patrimonio, 7553:persona, 8000:procedente,

9247:solar,

24:abolición

25:abolir, 98:acción, 673:anular, 2653:costumbre, 3565:efecto,
6037:ley,

2172:coma,

25:abolir

2653:costumbre, 5761:invalidar, 6037:ley,

24:abolición, 2991:derogar, 4151:estamento,

26:abonar

163:acreditar, 436:algo, 969:asistir, 971:asociación, 1107:avaluar,
1455:buen, 1564:calificar, 1656:cantidad, 2637:cosa, 3539: echar,
4499:fertilizante, 5623:inscribir, 6181:lugar, 6392:materia,
7226:pagar, 7229:pago, 7553:persona, 7744:poder, 8377:recibir,
9119:servicio, 9675:tierra, 10039:valor,

27:abono, 163:acreditar, 4500:fertilizar, 7342:pasaje,

27:abono

26:abonar, 98:acción, 572:añadir, 969:asistir, 1656:cantidad,
2439:constar, 2984:derecho, 3470:documento, 3565:efecto, 4036:ese,
5623:inscribir, 6181:lugar, 6453:mejor, 8377:recibir,
9119:servicio, 9447:sustancia, 9675:tierra,

4178:estiércol, 4499:fertilizante,

28:abordar

436:algo, 1001:asunto, 1037:atracar, 2774:cuestión, 1954:chocar,
3314:difícil, 3635:embarcación, 3636:embarcadero, 3691:empezar,
3697:emprender, 6839:nave, 9857:tratar,

3644:embestida, 8911:sadismo,

29:aborigen

4365:extensión, 4630:forma, 5015:habitante, 7156:originario,
7230:país, 7553:persona, 7970:primitivo, 9377:suelo,

7970:primitivo,

30:abortar

436:algo, 3634:embarazo, 4663:fracasar, 5024:hacer,
5728:interrupción, 8124:provocar, 9388:sufrir,

4508:feto,

31:abrasar

305:agitar, 436:algo, 816:ardor, 1401:brasa, 1561:caliente,
1580:calor, 2943:demasiado, 3477:dolor, 4011:escozor, 4701:frío,
7354:pasión, 7620:picor, 7694:planta, 8015:producir, 8221:quemar,

8459:reducir, 9002:secar, 9018:sed, 9094:sentir, 9447:sustancia,
10284:vivo,

813:arder, 921:asar, 1700:carbonizar, 5385:incendio, 8221:quemar,

32:abrasión
98:acción, 864:arrancar, 3129:desgastar, 3565:efecto,
3773:enérgico, 4698:fricción, 5809:irritación, 6469:membrana,
8015:producir, 8199:purgante, 9408:superficial,

33:abrasivo, 3531:dureza,

33:abrasivo
32:abrasión, 282:afilar, 450:alisar, 1327:blando, 8015:producir,
8167:pulir, 8297:raspar, 8544:relativo, 9121:servir,
9447:sustancia,

3282:diamante, 8432:rectificar,

34:abrazadera
35:abrazar, 937:asegurar, 2637:cosa, 7628:pieza, 9121:servir,

1365:boquilla, 4219:estribo,

35:abrazar
115:aceptar, 138:acoger, 274:afecto, 1409:brazo, 1859:ceñir,
3466:doctrina, 4209:estrechar, 8822:rodear, 9066:señal,

34:abrazadera,

36:abreviado
2634:corto, 3988:escaso, 8714:resumir,

38:abreviatura,

37:abreviar
107:acelerar, 157:acortar, 4059:espacio, 9670:tiempo,

200:adelantar, 8714:resumir,

38:abreviatura
36:abreviado, 4019:escritura, 7235:palabra, 8621:representación,

1213:bar, 1869:centímetro, 1985:cifra, 4240:etcétera, 6554:micra,
6571:milicia, 9072:seno, 9166:símbolo,

39:abrigar
274:afecto, 559:amparar, 5254:idea, 6233:mal, 7102:opinión,
8661:resguardar, 9604:tener, 9670:tiempo,

40:abrigo, 4896:gorro, 6293:manguito, 9517:tapar, 10183:vestido,

40:abrigo
39:abrigar, 560:amparo, 1822:cavidad, 2637:cosa, 2645:costa,
4367:exterior, 4701:frío, 4828:general, 5956:largo, 6181:lugar,
6291:manga, 6393:material, 6830:natural, 6839:nave, 7903:prenda,
8099:protección, 8101:proteger, 8670:resistente, 8810:roca,
9563:tela, 10003:utilizar, 10212:viento,

1163:bahía, 1668:capilla, 1676:capote, 1682:capucha, 2763:cuello,
4695:fresco, 4762:gabardina, 4804:garita, 5575:inhóspito,
9877:trenca, 10152:verdugo,

41:abrir
297:afuera, 436:algo, 699:apartar, 1500:cabeza, 1531:cajón,
1684:capullo, 1902:cerrado, 1906:cerrojo, 2210:comienzo,
2320:concreto, 2449:construir, 2605:correr, 2626:cortar,
2637:cosa, 2815:curva, 2830:dar, 3030:desarrollar, 3078:descorrer,
3086:descubrir, 3184:despegar, 3503:dos, 4059:espacio,
4363:extender, 4367:exterior, 4865:girar, 4900:gozne, 4994:guía,
5024:hacer, 5377:inaugurar, 5708:interior, 5789:ir, 5923:lado,
6053:libro, 6181:lugar, 6347:marco, 6410:mayor, 7040:oculto,
7139:organismo, 7359:paso, 7534:permitir, 7553:persona,
7594:pestillo, 7595:pétalo, 7725:pliego, 8149:público,
8154:puerta, 8214:quedar, 8270:rajar, 8293:rasgar, 8838:romper,
9053:semejante, 9096:separar, 9124:sesión, 9179:sincero,
9670:tiempo, 9724:tomar, 10089:vehículo, 10137:ver,

17:abertura, 19:abierto, 713:aperitivo, 714:apertura, 1178:balcón,
1283:bellota, 1679:cápsula, 1904:cerrar, 2598:corredera, 2830:dar,
3439:diurno, 3979:escape, 4136:establecer, 5152:hoja,
5377:inaugurar, 5397:incisivo, 6004:legumbre, 6127:llamar,
6135:llave, 6298:manilla, 7132:ordenanza, 7614:picaporte,
7825:postigo, 7842:pozo, 8270:rajar, 8963:sangría, 9698:tirador,
9895:trinchera, 10044:válvula, 10131:ventilador,

Anexo 3. Lista de las primitivas encontradas¹⁵

<i>FS</i>	<i>FE</i>	<i>Palabra<Id></i>	<i>FS</i>	<i>FE</i>	<i>Palabra<Id></i>
4	13	abajo<1>	21	19	acreditar<163>
19	5	abatir<12>	12	08	actividad<171>
21	35	abertura<17>	13	30	actor<174>
13	1	abnegación<21>	8	11	acumular<185>
8	5	abrigar<39>	22	20	adaptar<196>
5	6	abrochar<42>	9	3	adicto<216>
23	8	absoluto<47>	5	2	adivinanza<219>
8	8	aburrido<70>	17	25	adjetivo<221>
24	3	acacia<79>	13	16	admiración<229>
21	2	acanto<88>	7	2	adobar<232>
28	12	accidente<97>	12	1	adoquín<235>
32	21	aceite<102>	13	1	adormidera<239>
7	4	acentuación<110>	13	1	aerolito<260>
6	3	aceptable<113>	28	6	afectar<272>
14	7	acertar<121>	22	48	afecto<274>
7	2	acogida<139>	8	2	afianzar<278>
9	43	acompañar<146>	2	3	afiliado<283>
7	6	acontecer<150>	4	13	aflicción<290>
25	3	acotar<161>	6	7	ágil<304>

¹⁵ Resultados obtenidos corriendo el programa vis con el argumento -l y los archivos de entrada y salida (el que contiene los identificadores de las primitivas y el que tendrá las salidas -número de flechas salientes FS- y las entradas -número de flechas entrantes FE-, la palabra, y su identificador entre <>).

<i>FS</i>	<i>FE</i>	<i>Palabra<Id></i>	<i>FS</i>	<i>FE</i>	<i>Palabra<Id></i>
7	45	agradable<309>	6	1	anegar<593>
11	2	agregar<319>	23	7	ángel<602>
9	1	agrimonia<326>	15	3	anglicanismo<606>
49	72	agua<330>	32	14	animal<621>
15	5	aguantar<333>	5	2	anión<624>
6	25	agujero<346>	8	3	ano<627>
7	3	ahora<353>	8	4	anochecer<629>
5	22	aislado<361>	9	5	ansia<637>
9	42	ajeno<365>	20	9	antecedente<643>
3	1	ajonjolí<369>	1	3	antelación<646>
3	10	alabar<376>	12	2	antibiótico<655>
12	2	alarma<384>	13	8	antigüedad<663>
14	5	albedrío<389>	28	14	anular<673>
9	15	alboroto<395>	10	5	anverso<676>
6	4	alcalino<399>	28	05	aparato<684>
7	5	alcantarilla<402>	10	17	aparente<694>
8	13	alcohólico<408>	13	23	apartar<699>
14	8	aleación<414>	8	2	apelar<707>
16	11	alegre<418>	5	4	apetecer<715>
7	4	alfabeto<426>	11	9	aplastar<723>
21	1	algarroba<431>	13	37	apoyar<742>
19	19	alimentar<443>	8	15	aprender<750>
8	1	alineación<447>	7	1	apresto<754>
16	6	aliviar<453>	30	19	apretar<757>
17	3	almeja<462>	3	9	aproximar<770>
11	3	almohada<467>	38	11	apuntar<778>
5	1	almuédano<471>	8	10	arado<788>
8	44	alrededor<481>	29	7	araña<790>
7	1	altanero<483>	18	91	árbol<799>
12	23	alteración<486>	5	1	arete<822>
7	1	altisonante<493>	12	21	argumento<831>
21	44	altura<498>	19	08	arma<839>
9	18	alumno<507>	30	2	armadillo<841>
26	4	alzar<513>	12	11	armadura<844>
6	2	amabilidad<515>	9	29	armazón<847>
7	2	amaestrar<518>	24	17	armonía<849>
2	2	amargor<524>	24	13	arrancar<864>
12	9	ámbito<533>	7	14	arreglar<872>
13	3	amo<546>	13	2	arresto<880>
20	27	amor<551>	13	17	arrojar<888>
6	5	amparar<559>	17	2	artesano<898>
4	3	añadido<570>	22	36	articulación<900>
16	13	anchura<586>	17	14	artificio<906>

<i>FS</i>	<i>FE</i>	<i>Palabra<Id></i>	<i>FS</i>	<i>FE</i>	<i>Palabra<Id></i>
17	20	asamblea<920>	76	62	base<1246>
5	1	ascensor<928>	15	10	batalla<1255>
7	1	asesino<947>	5	1	bate<1258>
16	31	asiento<958>	4	2	bautizo<1264>
26	3	asimilar<964>	8	42	beneficio<1289>
18	13	asistir<969>	14	2	betún<1298>
4	2	asomo<980>	6	9	bicicleta<1302>
3	6	aspereza<985>	2	2	biliar<1310>
16	13	aspirar<989>	4	2	biografía<1314>
10	33	asunto<1001>	23	91	blanco<1325>
3	7	atado<1004>	21	1	blues<1335>
4	19	atar<1009>	27	3	bocadillo<1338>
12	20	atender<1015>	37	19	bola<1349>
16	2	atletismo<1031>	8	4	bordado<1367>
4	4	atreverse<1045>	2	1	borgoña<1374>
3	2	audacia<1060>	13	16	bóveda<1394>
16	3	audiencia<1063>	1	4	boxeador<1396>
8	2	auditorio<1068>	4	3	brasa<1401>
30	1	aurícula<1078>	33	2	brea<1410>
10	6	ausente<1082>	4	18	brillo<1422>
14	2	autómata<1092>	17	14	brotar<1441>
11	30	autor<1098>	4	97	buen<1455>
7	8	autoritario<1100>	15	1	bufón<1458>
4	4	avaluar<1107>	5	2	bullicio<1463>
6	9	avanzar<1111>	11	33	buque<1467>
25	93	ave<1116>	10	10	burla<1471>
3	14	averiguar<1126>	2	3	burocrático<1475>
4	17	ayudar<1140>	5	30	buscar<1479>
7	1	azada<1144>	6	3	cabal<1484>
18	5	azote<1150>	20	8	caballero<1490>
22	30	azúcar<1152>	29	44	caballo<1492>
10	2	bacilo<1159>	27	96	cabeza<1500>
5	3	badajo<1162>	2	1	cabrío<1509>
18	15	bajar<1169>	6	1	cachaza<1516>
11	8	balanza<1176>	25	19	cadena<1520>
20	13	baloncesto<1187>	44	48	caer<1525>
32	3	banca<1195>	58	22	caja<1528>
36	18	banda<1199>	22	4	calamar<1538>
13	1	banderilla<1202>	13	14	calcular<1551>
12	2	baquelita<1212>	12	3	caldo<1554>
2	1	barracón<1231>	15	4	cáliz<1567>
28	2	barrera<1236>	6	5	callar<1572>
18	18	barro<1240>	25	24	calzado<1587>

<i>FS</i>	<i>FE</i>	<i>Palabra<Id></i>	<i>FS</i>	<i>FE</i>	<i>Palabra<Id></i>
20	69	cambio<1595>	16	2	cerezo<1898>
18	5	campamento<1604>	3	12	cerrada<1901>
5	2	campesino<1609>	68	38	cerrar<1904>
8	2	canalizar<1617>	6	5	cerveza<1913>
6	5	cañería<1631>	17	2	cesta<1920>
1	2	cannabis<1639>	5	2	chaleco<1924>
30	7	cañón<1642>	8	3	checo<1938>
7	1	cantábrico<1648>	11	4	chocar<1954>
14	2	caolín<1659>	11	1	cíclico<1972>
24	88	capacidad<1661>	18	19	cielo<1978>
11	1	capilaridad<1667>	8	35	científico<1981>
34	12	capital<1671>	18	7	cifra<1985>
8	5	capricho<1677>	17	19	cilindro<1991>
15	6	capullo<1684>	27	18	cinta<2003>
23	1	carabinero<1687>	7	12	cintura<2005>
12	46	característica<169	18	36	circular<2011>
10	17	carbono<1701>	18	54	circunstancia<2018
21	12	cardinal<1707>	10	1	ciruelo<2021>
45	41	carga<1712>	3	2	citación<2026>
11	23	cargo<1718>	18	11	ciudadano<2032>
6	1	caritativo<1725>	18	10	clara<2039>
16	6	carnero<1730>	9	1	claroscuro<2045>
46	27	carrera<1739>	17	3	cláusula<2053>
17	8	cartón<1758>	2	1	claxon<2064>
25	3	casar<1764>	8	7	cliente<2067>
39	14	casco<1769>	13	4	club<2078>
14	5	casilla<1776>	34	6	cobra<2089>
12	3	castaña<1780>	22	1	coca<2094>
6	28	castigo<1787>	9	3	codo<2114>
11	3	cátedra<1801>	9	6	cofradía<2116>
13	81	categoría<1804>	10	6	cohesión<2123>
8	33	católico<1809>	12	6	colaborar<2133>
5	2	cavar<1820>	5	9	colegio<2144>
17	12	cazar<1826>	8	6	colgado<2148>
13	3	cefalópodo<1833>	14	5	colmillo<2156>
14	1	celda<1837>	13	68	colocar<2159>
5	9	celeste<1843>	24	17	color<2164>
18	1	cenizo<1861>	37	31	columna<2170>
8	9	censurar<1866>	10	18	combate<2177>
23	37	central<1872>	2	1	combinatorio<2183>
4	11	cercano<1885>	15	5	comedia<2186>
8	12	cereal<1891>	6	30	comercial<2197>
22	5	cerebro<1894>	6	22	cometer<2204>

<i>FS</i>	<i>FE</i>	<i>Palabra<Id></i>	<i>FS</i>	<i>FE</i>	<i>Palabra<Id></i>
8	1	comisaría<2211>	3	17	continuar<2489>
14	6	cómodo<2217>	22	8	contracción<2495>
5	15	comparar<2222>	10	16	contraer<2498>
24	7	compás<2229>	9	10	contrariedad<2505>
15	7	competencia<2238>	8	27	contrato<2514>
7	6	competir<2242>	7	10	convencer<2525>
20	7	complejo<2248>	5	28	conveniente<2530>
9	38	completo<2253>	4	3	convidar<2541>
10	7	compostura<2263>	4	8	convocar<2545>
13	76	comprender<2268>	30	12	copa<2553>
5	13	comprobar<2276>	14	1	copo<2559>
20	13	compromiso<2279>	12	1	corchete<2567>
28	02	común<2284>	8	3	córnea<2578>
18	49	comunidad<2288>	34	8	corona<2583>
6	3	concejal<2294>	9	1	coronilla<2585>
16	7	concertar<2301>	3	1	corpuscular<2591>
18	13	conciencia<2304>	21	1	corredera<2598>
4	4	concluir<2312>	22	7	correlación<2602>
9	4	concurrir<2325>	17	3	corro<2614>
9	3	condensación<2331>	13	1	curso<2622>
3	2	condescender<2334>	5	16	cortante<2625>
10	25	conducir<2342>	3	14	cortés<2630>
8	5	confección<2350>	18	34	corteza<2632>
18	3	confesión<2355>	16	1	cosa<2637>
9	21	confianza<2357>	9	1	cosquilleo<2644>
17	4	confirmación<2365>	15	10	costado<2646>
6	20	conformidad<2374>	21	57	costumbre<2653>
18	6	congreso<2385>	25	2	coto<2658>
7	2	conjugarse<2389>	15	9	creer<2677>
29	2	conjuntivo<2392>	12	3	cresta<2683>
9	2	conmoción<2397>	7	2	criterio<2702>
43	8	cono<2400>	12	1	crucera<2713>
18	72	conocimiento<2404>	10	3	cuadernillo<2723>
23	11	consagrar<2408>	3	9	cuadrúpedo<2732>
8	6	consentimiento<241	5	12	cualidad<2737>
14	5	conservador<2422>	16	27	cubierto<2754>
9	20	consideración<2425	3	1	cuchufleta<2761>
4	3	consolar<2433>	19	59	cuenta<2765>
22	13	constitución<2443>	32	74	cuerda<2767>
3	3	constitutivo<2445>	6	23	cuero<2770>
5	5	consumido<2457>	10	4	cuidadoso<2778>
22	29	contacto<2462>	19	31	cultivar<2787>
5	1	contemplativo<2471	33	21	culto<2789>

<i>FS</i>	<i>FE</i>	<i>Palabra<Id></i>	<i>FS</i>	<i>FE</i>	<i>Palabra<Id></i>
4	19	cultural<2791>	19	2	desfigurar<3123>
9	5	cúpula<2803>	7	14	desgraciado<3132>
11	12	curtir<2814>	9	41	designar<3140>
4	2	danzante<2829>	5	4	desinteresado<3150>
17	4	debate<2834>	4	1	desmejorar<3160>
8	19	debe<2836>	36	1	despejar<3187>
8	15	débil<2839>	15	5	desplazar<3198>
23	30	decir<2855>	13	33	destacar<3218>
30	21	declarar<2859>	20	7	destilar<3225>
25	4	declinar<2861>	14	5	desviación<3244>
9	9	decoración<2863>	10	14	detener<3254>
13	62	dedicar<2873>	5	4	devanar<3266>
11	24	defecto<2880>	8	3	devoto<3271>
5	4	defensor<2884>	20	4	diablo<3273>
8	9	definido<2890>	16	8	diálogo<3281>
20	4	degradar<2902>	13	46	dibujo<3289>
6	31	delante<2908>	8	6	dictado<3295>
26	56	delgado<2919>	20	4	diferencial<3310>
14	19	delicado<2924>	2	13	difundir<3318>
25	11	demonstración<2946>	16	15	digestivo<3324>
5	4	dentadura<2956>	16	28	dimensión<3337>
12	71	deporte<2970>	10	2	dinamismo<3341>
14	23	depositar<2975>	15	99	dinero<3343>
25	13	depresión<2979>	5	4	dióxido<3347>
64	07	derecho<2984>	17	3	díptero<3350>
10	2	dermis<2990>	6	5	directivo<3354>
13	7	derrotar<3000>	31	73	dirigir<3358>
8	1	desafecto<3005>	17	13	disciplina<3361>
3	4	desagradar<3009>	6	10	discurrir<3377>
8	16	desaparecer<3025>	12	13	discutir<3380>
32	54	desarrollar<3030>	2	5	disimulado<3392>
6	2	desatar<3036>	19	15	disolución<3400>
22	16	descansar<3048>	16	17	disolver<3403>
3	7	descarado<3051>	10	30	disponer<3412>
15	2	descartar<3055>	7	2	distender<3422>
6	2	desconcertar<3067>	27	42	distinguir<3426>
7	9	desconocido<3073>	5	12	distinto<3428>
4	15	descubierto<3084>	5	23	diversión<3442>
11	23	descubrir<3086>	26	15	doblar<3456>
11	2	descuidar<3089>	8	10	dogma<3471>
6	2	desdoblar<3098>	19	61	dolor<3477>
16	7	desembocar<3105>	17	35	dominar<3486>
6	4	desengaño<3110>	7	92	dos<3503>

<i>FS</i>	<i>FE</i>	<i>Palabra<Id></i>	<i>FS</i>	<i>FE</i>	<i>Palabra<Id></i>
27	25	droga<3514>	8	12	entregar<3878>
14	32	dulce<3524>	22	2	entrevista<3886>
8	13	dureza<3531>	11	10	entusiasmo<3890>
53	51	echar<3539>	16	5	enunciado<3895>
7	2	ectodermo<3547>	20	21	envolver<3914>
9	1	edicto<3553>	14	37	época<3924>
8	7	edificar<3555>	3	1	equilibrado<3928>
19	61	efecto<3565>	12	2	equino<3932>
2	1	efervescente<3568>	21	29	equipo<3938>
12	2	egoísmo<3578>	11	24	equivaler<3942>
14	11	ejemplar<3584>	9	9	equivocar<3944>
4	15	elaboración<3590>	13	5	escabroso<3962>
9	13	electricidad<3598>	14	15	escalera<3967>
2	6	elegancia<3609>	18	1	escarabajo<3981>
8	15	elegir<3612>	8	7	escarpado<3985>
3	26	elevado<3616>	6	5	esclavo<3995>
9	2	embarcar<3637>	38	3	escobilla<3997>
7	7	emblema<3646>	10	10	esconder<4006>
3	1	embravecer<3651>	24	1	escribano<4013>
19	7	embutido<3659>	8	84	escrito<4016>
9	11	emisor<3669>	8	5	escrúpulo<4020>
14	4	empapar<3680>	21	20	escultura<4034>
8	3	enamorar<3709>	25	16	esencia<4037>
18	3	encabezar<3711>	21	1	esguince<4046>
10	5	encaminar<3715>	25	5	esmalte<4051>
4	1	encarcelar<3720>	6	3	esófago<4057>
9	11	encargar<3724>	2	3	espacioso<4060>
19	5	encerado<3735>	39	34	español<4066>
23	1	enciclopedia<3739>	14	7	especialista<4077>
5	1	encubridor<3753>	10	56	espectáculo<4082>
18	2	endodermo<3765>	30	9	espectro<4085>
5	6	enfrente<3787>	4	7	esperanza<4090>
3	13	engendrar<3799>	28	43	espíritu<4109>
10	3	enhebrar<3810>	4	9	esplendor<4112>
11	15	enlazar<3815>	18	21	esqueleto<4129>
9	5	enmendar<3819>	16	4	estabilidad<4134>
20	5	enredar<3826>	8	1	estacionamiento<4138>
10	2	enrejado<3828>	33	51	estado<4146>
10	2	ensamblar<3836>	13	4	estambre<4150>
14	23	enseñar<3849>	29	7	estancia<4157>
4	3	entendido<3858>	3	1	esterlina<4173>
14	9	enterrar<3865>	19	2	estética<4175>
45	36	entrar<3875>	39	36	estilo<4183>

<i>FS</i>	<i>FE</i>	<i>Palabra<Id></i>	<i>FS</i>	<i>FE</i>	<i>Palabra<Id></i>
7	15	estimación<4186>	7	1	febrero<4465>
11	21	estímulo<4190>	11	2	fécula<4468>
13	18	estómago<4197>	14	3	federación<4473>
14	1	estrado<4202>	16	1	fenicio<4487>
20	10	estrofa<4222>	17	12	ferrocarril<4496>
43	57	estudio<4234>	6	3	fervoroso<4502>
15	14	etapa<4239>	13	18	fibra<4512>
8	2	evaluación<4255>	9	1	fichero<4518>
9	2	evidencia<4263>	21	13	fiel<4524>
3	37	evitar<4265>	19	31	fiesta<4528>
6	8	evolucionar<4268>	20	38	fijar<4534>
3	25	exacto<4271>	10	8	filamento<4537>
13	26	examen<4278>	36	22	filosofía<4545>
7	6	excelencia<4284>	3	12	finalidad<4551>
9	32	exceso<4291>	8	23	fingir<4557>
11	2	excusa<4303>	29	9	firma<4560>
6	5	exhalar<4308>	14	2	fiscal<4565>
15	23	exigir<4313>	11	98	físico<4569>
6	16	existencia<4315>	11	18	flexible<4584>
13	3	exoesqueleto<4318>	11	6	flojo<4586>
10	4	expedir<4327>	5	2	fluidez<4601>
17	22	experimentar<4332>	29	6	foco<4607>
7	10	experto<4334>	44	40	fondo<4621>
10	13	explicación<4338>	11	9	fonético<4623>
43	20	exposición<4353>	20	18	formación<4631>
7	48	expreso<4358>	11	1	formalismo<4634>
5	8	exquisito<4361>	7	7	fortificar<4647>
19	66	exterior<4367>	11	3	fósil<4655>
7	10	extinguir<4369>	20	6	fracción<4665>
6	5	extrañeza<4377>	6	11	franja<4680>
14	19	extremidad<4386>	14	17	frecuencia<4687>
10	36	fabricar<4394>	6	3	frialdad<4697>
8	62	fácil<4400>	3	2	frondosidad<4706>
6	13	facilitar<4402>	13	72	fuego<4720>
27	65	facultad<4406>	31	1	fuero<4724>
15	2	falsear<4418>	19	59	fuerza<4726>
20	17	falta<4423>	30	15	función<4734>
28	51	familia<4427>	13	20	fundamento<4743>
12	13	famoso<4430>	23	4	fundir<4747>
30	11	fantasía<4436>	20	1	gaita<4765>
17	17	fase<4450>	11	1	galactosa<4768>
4	26	favorable<4460>	28	7	galería<4776>
11	3	favorito<4463>	6	2	gameto<4788>

<i>FS</i>	<i>FE</i>	<i>Palabra<Id></i>	<i>FS</i>	<i>FE</i>	<i>Palabra<Id></i>
17	23	ganar<4795>	15	6	hermano<5081>
24	8	garantía<4798>	8	7	hermoso<5083>
15	3	gasa<4809>	15	3	hervir<5091>
7	17	gastar<4815>	17	1	hibernación<5098>
12	4	gelatina<4821>	31	48	hierro<5111>
12	2	gen<4825>	17	3	higiene<5113>
15	96	general<4828>	8	1	hilado<5119>
33	14	género<4834>	25	64	hilo<5122>
16	11	genio<4839>	17	3	hipótesis<5138>
10	50	gente<4841>	5	23	histórico<5144>
17	3	germano<4852>	10	5	hocico<5149>
4	2	gigantesco<4860>	5	1	hojalata<5153>
34	2	gitano<4868>	17	3	homenaje<5160>
10	27	gobernar<4882>	3	1	
21	54	golpe<4890>			homogeneidad<5163>
5	4	gordura<4894>	6	2	honesto<5172>
8	6	grabación<4902>	5	14	honrado<5179>
13	33	gracia<4905>	10	33	horizontal<5185>
20	3	gradación<4908>	11	2	hormigón<5189>
60	74	grado<4910>	20	17	horno<5194>
17	24	gráfico<4916>	3	4	hospedar<5204>
14	07	gramática<4918>	13	12	hotel<5212>
13	3	gramíneo<4920>	9	6	huerta<5220>
10	61	grande<4925>	21	30	huevo<5224>
22	18	grasa<4936>	7	10	humedad<5233>
6	6	gratitud<4940>	5	5	humillación<5238>
9	10	gravedad<4947>	24	9	humor<5241>
9	36	grueso<4967>	5	2	hundimiento<5245>
18	11	guarda<4975>	6	3	ida<5253>
14	10	guardia<4979>	14	14	ideal<5255>
31	12	guía<4994>	8	8	idear<5259>
24	3	guión<4998>	21	10	identificar<5262>
8	33	habilidad<5009>	6	1	idílico<5265>
13	19	habla<5020>	6	5	idóneo<5269>
55	45	hacer<5024>	6	6	ignorancia<5272>
10	12	hacienda<5027>	19	70	igual<5275>
10	38	hallar<5032>	23	19	igualdad<5278>
10	18	harina<5038>	20	4	ilustrar<5290>
6	4	hebreo<5046>	31	62	imagen<5292>
18	66	hecho<5050>	12	13	imitación<5299>
23	39	hembra<5063>	5	1	impaciente<5302>
13	1	hemoglobina<5066>	9	3	imperativo<5313>
22	3	heredar<5072>	13	1	impertinente<5323>

<i>FS</i>	<i>FE</i>	<i>Palabra<Id></i>	<i>FS</i>	<i>FE</i>	<i>Palabra<Id></i>
7	66	importancia<5332>	14	12	intermedio<5713>
16	14	importar<5334>	18	3	interrogativo<5725>
17	39	impresión<5345>	22	31	intervenir<5735>
26	30	imprimir<5353>	6	2	intoxicación<5741>
6	26	impuesto<5360>	4	2	intrínseco<5750>
10	8	impureza<5365>	9	2	intuir<5755>
5	3	inadvertido<5372>	12	14	investigación<5772>
4	2	inaudito<5375>	7	2	inviolable<5779>
10	2	incapacitar<5381>	8	3	involucrar<5784>
8	8	incisivo<5397>	30	71	ir<5789>
11	41	incluir<5402>	17	4	iris<5791>
2	4	incomodar<5408>	22	18	irregular<5800>
6	1	inconsecuente<5421>	5	3	islámico<5814>
5	2	incrustación<5434>	14	3	izquierdo<5821>
6	1	incursión<5442>	10	12	jardín<5836>
23	7	indefinido<5450>	17	21	jefe<5843>
19	26	independiente<5455>	17	1	jeringuilla<5848>
10	94	indicar<5459>	14	3	joroba<5857>
35	5	índice<5461>	13	17	joya<5860>
11	4	indirecto<5471>	38	33	juego<5867>
10	14	individual<5479>	25	24	jugar<5873>
5	1	indonesio<5487>	21	67	juicio<5878>
20	27	industria<5497>	4	35	junto<5885>
10	1	infectar<5519>	9	21	justo<5898>
17	6	inferioridad<5522>	11	27	labor<5907>
13	7	inflamación<5534>	20	6	labrar<5914>
10	26	influencia<5540>	14	24	lámina<5937>
5	5	infringir<5554>	10	19	lana<5941>
7	1	ingeniero<5559>	18	2	langostino<5945>
10	7	iniciativa<5581>	10	33	larga<5955>
5	1	inmejorable<5591>	24	5	laringe<5958>
14	4	inorgánico<5615>	21	7	latino<5972>
10	7	insensible<5630>	8	2	lavado<5977>
9	1	inspector<5649>	10	4	lealtad<5982>
7	24	instalación<5652>	9	27	leche<5985>
5	7	instituir<5659>	29	6	lector<5988>
14	76	instrumento<5665>	7	30	legal<5993>
4	3	insultar<5669>	3	11	lejano<6006>
22	20	inteligencia<5682>	40	12	lengua<6011>
9	35	intención<5687>	19	1	lepidóptero<6020>
8	19	intentar<5692>	6	3	letrero<6026>
11	5	intercambio<5697>	48	98	ley<6037>
23	59	interés<5702>	17	26	libertad<6046>

<i>FS</i>	<i>FE</i>	<i>Palabra<Id></i>	<i>FS</i>	<i>FE</i>	<i>Palabra<Id></i>
23	83	libro<6053>	10	4	marfil<6350>
9	5	licenciado<6055>	23	60	masa<6373>
9	7	lidia<6061>	2	2	masón<6380>
26	2	ligar<6068>	19	13	material<6393>
40	1	lima<6073>	20	11	matiz<6398>
14	5	limón<6080>	21	14	matrimonio<6404>
9	12	linaje<6087>	13	14	máxima<6408>
48	95	línea<6091>	24	2	maza<6414>
23	3	liquidar<6103>	7	42	mecanismo<6420>
15	18	liso<6108>	17	8	medalla<6428>
4	2	litigante<6116>	8	1	medianoche<6433>
4	4	llaga<6123>	14	73	medida<6441>
11	14	llano<6131>	10	17	mejorar<6454>
31	93	llegar<6137>	12	1	melena<6459>
37	33	llevar<6141>	7	5	melodía<6465>
9	20	lluvia<6144>	16	51	menor<6479>
26	3	loco<6152>	13	2	menta<6488>
7	7	lógico<6161>	8	11	mentira<6492>
6	3	lombriz<6164>	11	10	menudo<6496>
14	34	longitud<6167>	1	3	merecedor<6503>
6	9	luchar<6178>	17	21	mesa<6513>
31	99	luz<6195>	11	2	metabolismo<6523>
6	12	maduro<6212>	7	1	metamórfico<6530>
14	1	mafia<6216>	17	37	mezcla<6548>
10	3	magma<6220>	3	1	miau<6552>
14	18	magnitud<6229>	9	13	miedo<6560>
9	2	majestad<6231>	13	77	militar<6573>
4	1	malentendido<6241>	12	6	ministerio<6587>
51	25	malo<6255>	24	3	misterio<6604>
22	32	mancha<6272>	5	7	mitigar<6608>
6	4	manchar<6274>	2	4	mobiliario<6611>
10	9	mandato<6281>	12	4	moco<6613>
15	1	mandil<6283>	23	26	modelo<6618>
13	12	mando<6285>	6	8	moderar<6622>
7	5	manejo<6289>	22	1	modulación<6631>
12	20	mango<6292>	15	6	mojar<6639>
9	14	manifiesto<6297>	4	5	molecular<6646>
4	2	mañoso<6309>	15	27	molusco<6653>
12	1	manubrio<6320>	14	9	monarquía<6657>
18	4	mapa<6325>	7	15	monetario<6664>
20	80	máquina<6328>	7	2	monopolizar<6670>
54	13	marcar<6341>	26	11	montar<6681>
18	12	marco<6347>	11	5	moreno<6702>

<i>FS</i>	<i>FE</i>	<i>Palabra<Id></i>	<i>FS</i>	<i>FE</i>	<i>Palabra<Id></i>
7	30	mostrar<6721>	5	16	ocasionar<7023>
18	65	mover<6732>	20	6	occidente<7026>
6	90	mucho<6742>	4	4	ocultación<7036>
19	5	mudar<6748>	7	1	ocultista<7039>
19	7	muela<6751>	15	61	ocupar<7042>
5	1	mundano<6771>	5	5	ofensivo<7053>
13	4	munición<6776>	32	34	oficial<7055>
6	8	muralla<6780>	12	60	oficio<7057>
23	43	música<6793>	17	29	ofrecer<7059>
22	29	nacer<6805>	28	38	ojo<7067>
19	11	nacimiento<6807>	6	1	olivo<7074>
13	2	naif<6814>	6	4	olvido<7080>
15	6	naranja<6818>	7	1	ondear<7088>
26	16	nariz<6821>	9	79	opinión<7102>
3	9	narrar<6823>	3	12	oportunidad<7106>
32	25	natural<6830>	16	16	oral<7120>
20	25	nave<6839>	42	57	orden<7126>
5	12	navegar<6843>	20	11	ordenador<7130>
14	28	necesidad<6847>	13	37	organización<7140>
7	5	negociación<6858>	23	23	órgano<7143>
17	42	negro<6861>	12	10	orientar<7149>
8	24	nervioso<6868>	34	17	original<7153>
6	39	niño<6889>	12	3	ortografía<7171>
3	2	nitrate<6893>	18	5	oscurecer<7182>
12	04	no<6900>	7	7	ovario<7198>
19	92	nombre<6914>	30	9	paciencia<7211>
18	74	normal<6921>	18	18	padre<7218>
8	1	nostalgia<6928>	11	5	pagano<7225>
10	13	notable<6930>	10	9	página<7228>
5	2	novato<6938>	39	14	palabra<7235>
43	5	nudo<6955>	12	1	palco<7242>
21	36	nuevo<6960>	8	6	pálido<7247>
5	5	nulo<6962>	26	4	palma<7250>
24	43	número<6967>	24	49	palo<7256>
4	11	obispo<6980>	34	26	pan<7262>
4	5	objeción<6982>	11	2	panza<7273>
21	05	objeto<6985>	6	1	papelera<7277>
31	74	obligación<6988>	9	5	parado<7287>
17	28	obra<6993>	29	13	paralelo<7293>
17	4	obrero<6995>	15	50	pared<7304>
13	8	observación<7000>	19	4	paréntesis<7308>
3	2	obstaculizar<7007>	12	6	parroquia<7320>
4	7	obtención<7015>	6	32	participar<7324>

<i>FS</i>	<i>FE</i>	<i>Palabra<Id></i>	<i>FS</i>	<i>FE</i>	<i>Palabra<Id></i>
45	20	partida<7330>	6	2	picazón<7618>
31	3	pasaje<7342>	54	81	pie<7622>
81	93	pasar<7347>	23	86	piel<7625>
37	46	pasión<7354>	33	27	pieza<7628>
70	71	paso<7359>	4	1	piloso<7636>
25	21	pasta<7361>	15	3	piña<7640>
27	3	pastilla<7366>	28	4	pisar<7665>
19	7	patente<7374>	29	2	pista<7669>
22	2	pato<7383>	12	3	pistón<7672>
33	15	pausa<7394>	37	3	pizarra<7675>
22	5	paz<7402>	4	1	placentero<7678>
3	22	peculiar<7408>	20	21	plan<7682>
25	3	pedernal<7415>	22	11	planeta<7689>
14	8	pedúnculo<7421>	21	11	plato<7711>
14	18	pelea<7431>	17	8	pleito<7721>
8	15	peligroso<7435>	10	13	pliegue<7726>
8	1	peluquero<7441>	21	1	plumón<7731>
12	3	penal<7447>	12	7	pobre<7739>
20	17	pendiente<7453>	34	90	poder<7744>
16	2	penitenciario<7465>	10	13	poema<7749>
23	29	pensamiento<7469>	15	18	poético<7753>
4	3	penúltimo<7474>	39	10	polo<7770>
10	95	pequeño<7482>	9	6	pólvora<7773>
5	6	perceptible<7487>	71	55	poner<7779>
16	13	percusión<7489>	12	25	popular<7783>
12	24	pérdida<7492>	7	61	porción<7787>
7	10	perdonar<7496>	12	7	porte<7799>
12	18	perfección<7503>	9	2	portero<7801>
33	6	perfil<7506>	21	15	posesión<7810>
9	3	perfumar<7510>	13	22	posibilidad<7813>
14	5	periodista<7521>	4	1	posteridad<7823>
2	16	perjuicio<7527>	16	13	postura<7832>
4	11	permanente<7531>	28	16	potencia<7835>
5	2	perplejidad<7544>	13	4	potente<7839>
7	1	perseverancia<7548>	9	27	practicar<7845>
14	8	personalidad<7556>	9	21	preceder<7853>
4	2	perspicacia<7561>	13	18	preciso<7864>
22	8	pesa<7577>	4	5	predecir<7869>
17	10	pesca<7583>	3	3	predilección<7878>
28	27	pez<7601>	5	1	prehistórico<7892>
14	5	piano<7606>	3	4	premiar<7899>
10	2	picador<7610>	18	70	prenda<7903>
84	8	picar<7615>	18	9	preocupación<7908>

<i>FS</i>	<i>FE</i>	<i>Palabra<Id></i>	<i>FS</i>	<i>FE</i>	<i>Palabra<Id></i>
9	2	prepucio<7915>	10	2	radiodifusión<8265>
10	2	prescripción<7920>	11	3	rajar<8270>
27	21	presente<7926>	6	50	rápido<8283>
14	10	presumir<7944>	12	6	raro<8288>
16	13	prevenir<7958>	19	3	raso<8296>
9	21	previo<7960>	11	2	ratero<8304>
17	17	principal<7975>	31	17	rayo<8312>
23	76	principio<7979>	25	79	razón<8314>
13	9	prisión<7982>	33	24	reacción<8320>
7	22	privado<7986>	50	1	realismo<8330>
24	16	privar<7988>	11	2	rebaja<8334>
21	19	probar<7996>	21	4	rebelde<8342>
13	1	procesado<8003>	7	3	reborde<8345>
6	2	procura<8010>	11	7	recaer<8354>
19	74	producto<8017>	15	4	recepción<8366>
25	35	profundo<8031>	14	14	rechazar<8372>
11	4	progreso<8038>	8	40	recipiente<8383>
11	9	prominencia<8051>	7	2	reclusión<8393>
5	7	prontitud<8060>	67	39	recoger<8400>
9	12	propagar<8067>	16	10	recompensa<8406>
23	60	propiedad<8072>	12	1	reconversión<8416>
17	16	proponer<8076>	33	3	rectificar<8432>
27	38	proposición<8081>	25	25	recto<8435>
2	5	proseguir<8092>	20	1	recuperación<8439>
15	1	protesto<8110>	23	16	recurso<8442>
11	13	provincia<8119>	10	2	redactor<8446>
5	1	provocativo<8125>	15	25	redondo<8456>
22	17	proyectar<8129>	14	30	referir<8465>
42	47	prueba<8134>	34	11	reflexión<8472>
3	22	psíquico<8141>	5	2	refrenar<8479>
14	14	publicar<8145>	7	12	regalo<8492>
12	62	pueblo<8152>	37	10	régimen<8501>
10	12	puerto<8155>	15	24	regir<8506>
7	2	pulcritud<8163>	24	55	regla<8510>
5	18	pulmón<8168>	12	1	rehacer<8520>
6	1	pulque<8173>	9	1	reinar<8525>
13	9	puño<8184>	20	7	reír<8528>
6	5	pureza<8197>	10	1	rejón<8534>
21	3	quebrantar<8211>	5	5	relatar<8543>
11	7	queja<8216>	4	1	relevancia<8546>
41	6	quemar<8221>	15	18	relieve<8550>
3	2	quieto<8231>	8	3	rellano<8554>
14	54	quitar<8245>	9	2	remite<8569>

<i>FS</i>	<i>FE</i>	<i>Palabra<Id></i>	<i>FS</i>	<i>FE</i>	<i>Palabra<Id></i>
12	7	remo<8572>	19	12	rumiante<8878>
16	4	rendir<8586>	6	3	sábado<8888>
14	20	reparar<8597>	7	7	sabiduría<8891>
6	3	replicar<8609>	8	11	sabroso<8897>
11	13	reposo<8617>	13	24	sacerdote<8901>
22	20	representar<8623>	13	11	saco<8903>
12	33	reproducir<8631>	12	3	sacro<8907>
10	10	repugnancia<8637>	31	28	sal<8915>
3	3	resarcir<8649>	8	4	saldar<8922>
7	6	resentimiento<8655>	5	21	saliente<8927>
21	7	reservar<8659>	14	2	saliva<8931>
8	8	residir<8664>	8	3	saludable<8943>
16	9	resina<8668>	9	1	salvación<8947>
11	22	resolución<8672>	5	2	sanar<8957>
10	2	resonancia<8674>	2	1	sanguinario<8967>
4	3	respetable<8681>	31	25	santo<8974>
8	42	respeto<8683>	10	3	sargento<8981>
16	4	respirar<8686>	28	1	satélite<8990>
5	6	resplandor<8689>	18	1	saurio<8995>
7	14	respuesta<8694>	8	4	sazón<8997>
17	7	restar<8700>	39	24	sección<9003>
14	27	resto<8704>	15	15	sector<9012>
4	3	restringir<8706>	10	10	secundario<9017>
8	41	resulta<8709>	16	9	seducir<9026>
18	13	resultar<8712>	16	11	segmento<9030>
19	10	retirar<8723>	3	11	seguidor<9033>
8	3	retoño<8726>	5	2	selección<9039>
13	5	retrasar<8736>	14	10	semana<9046>
9	1	retroproyector<874>	13	18	semejanza<9054>
9	2	reverencia<8752>	8	1	semitono<9062>
9	2	revocar<8761>	27	70	señal<9066>
12	13	rígido<8783>	24	9	sencillo<9070>
2	14	riña<8788>	14	41	sensación<9076>
17	3	riñón<8790>	6	4	sentado<9086>
17	15	ritmo<8796>	35	14	sentido<9091>
12	9	roce<8811>	10	11	separación<9095>
16	3	rodado<8817>	13	4	sepulcro<9099>
11	7	rodilla<8825>	8	2	séquito<9104>
23	6	románico<8833>	12	4	sereno<9108>
21	16	romper<8838>	17	9	serio<9111>
27	3	rosal<8848>	8	8	severo<9127>
8	6	rotura<8859>	16	4	sierra<9135>
2	4	ruego<8868>	8	38	siglo<9139>

<i>FS</i>	<i>FE</i>	<i>Palabra<Id></i>	<i>FS</i>	<i>FE</i>	<i>Palabra<Id></i>
23	41	significado<9142>	4	3	suspica<9445>
4	10	siguiente<9147>	25	25	sustantivo<9450>
9	1	silogismo<9163>	3	31	sustituir<9454>
6	1	simpatizante<9173>	15	14	tabaco<9461>
8	1	síndrome<9181>	51	22	tabla<9466>
9	9	sintagma<9187>	42	9	taco<9477>
3	1	sistemática<9197>	7	1	tala<9486>
7	10	sitio<9199>	12	3	talonario<9500>
14	12	situat<9202>	19	8	tan<9506>
3	1	sobrado<9209>	19	14	tapa<9515>
6	47	sobresalir<9220>	17	7	tapiz<9521>
6	1	socialista<9230>	19	46	teatro<9547>
35	43	sol<9245>	3	3	tedio<9556>
8	39	soldado<9251>	13	90	tejido<9561>
9	1	solera<9260>	12	11	tela<9563>
5	1	solidario<9263>	8	30	televisión<9574>
19	48	sólo<9269>	8	4	temer<9583>
8	1	solvente<9278>	4	11	temor<9585>
5	15	sombrero<9282>	22	17	temporal<9594>
24	47	someter<9285>	7	1	tenderete<9600>
17	15	sonoro<9296>	17	1	tenor<9608>
9	10	sorprender<9311>	12	25	teoría<9620>
3	1	sosegar<9319>	2	1	terminante<9632>
9	5	soso<9321>	18	14	terreno<9645>
10	9	subasta<9333>	14	14	testamento<9656>
5	7	súbito<9340>	9	42	texto<9665>
13	4	subjuntivo<9342>	25	48	tiempo<9670>
5	1	subsanan<9352>	5	4	tienta<9672>
16	67	sucedat<9361>	9	2	tilde<9678>
4	12	suciedad<9366>	11	2	tino<9686>
16	5	sudor<9372>	14	3	tirada<9696>
25	49	suelo<9377>	19	1	tirano<9701>
13	18	sueño<9379>	25	14	tiro<9705>
5	49	sujetar<9391>	5	2	tocadiscos<9714>
7	4	sumisión<9403>	4	22	todo<9719>
2	6	suntuoso<9405>	22	17	toma<9723>
6	18	superficial<9408>	36	28	tono<9732>
17	10	superioridad<9412>	12	4	tonto<9735>
12	3	súplica<9420>	8	10	toque<9740>
11	3	suplir<9423>	28	2	tordo<9745>
10	18	sur<9431>	14	5	tormento<9751>
9	5	susceptible<9439>	9	18	torpe<9757>
25	21	suspender<9441>	6	3	tostada<9769>

<i>FS</i>	<i>FE</i>	<i>Palabra<Id></i>	<i>FS</i>	<i>FE</i>	<i>Palabra<Id></i>
2	5	tóxico<9775>	16	5	vejiga<10096>
18	15	trabajo<9781>	12	1	velero<10100>
17	8	tradición<9786>	21	6	velo<10104>
18	6	traducción<9788>	42	5	vena<10108>
7	5	traficar<9792>	21	25	venir<10124>
6	4	traicionar<9801>	32	21	verbal<10143>
7	8	tranquilo<9814>	16	36	verdad<10147>
6	31	transformar<9822>	5	4	verdor<10150>
8	9	tránsito<9827>	29	42	verso<10171>
26	21	transparente<9834>	18	6	verter<10175>
10	11	trasladar<9847>	7	5	vestidura<10184>
22	74	tratar<9857>	32	31	vestir<10187>
13	1	travesía<9861>	18	23	vía<10191>
10	14	trazar<9869>	15	7	vibrar<10197>
14	3	trepar<9883>	25	16	vicio<10201>
7	35	tribunal<9890>	10	5	vid<10205>
9	3	triturar<9903>	8	9	vigilancia<10218>
11	2	trompeta<9910>	14	5	vinagre<10225>
32	27	tronco<9912>	8	39	vino<10229>
11	7	tropezar<9916>	11	33	violencia<10235>
3	5	tubular<9929>	15	1	viril<10243>
17	2	tulipa<9931>	10	2	virulento<10249>
4	5	turbación<9940>	13	1	visera<10255>
7	55	último<9954>	38	52	vista<10265>
14	46	unión<9966>	15	18	viveza<10279>
4	6	universo<9973>	17	90	vivo<10284>
6	6	urbano<9979>	11	4	voluble<10299>
7	5	urgencia<9984>	29	65	voluntad<10302>
3	71	usado<9990>	50	49	volver<10308>
8	49	utensilio<9997>	9	12	voto<10315>
6	17	utilidad<10000>	29	25	vuelta<10318>
10	4	vacilar<10011>	7	1	yeyuno<10331>
15	4	vago<10017>	19	61	zona<10350>
3	1	valedor<10024>			
5	6	valía<10030>			
5	6	vallado<10037>			
10	9	valorar<10041>			
8	23	vapor<10050>			
24	4	variante<10057>			
29	31	variedad<10059>			
7	1	varonil<10064>			
9	1	vecindad<10077>			
8	52	vegetal<10084>			

Anexo 4. Lista de los ciclos encontrados¹⁶

<i>LC</i>	<i>FS</i>	<i>FE</i>	<i>Ciclo</i>
2	17	3	ábaco<0> -> dórico<3495> -> ábaco<0>
2	4	13	abajo<1> -> bajo<1170> -> abajo<1>
3	3	3	abastecer<9> -> proveer<8116> -> suministrar<9401> -> abastecer<9>
2	19	5	abatir<12> -> humillar<5239> -> abatir<12>
2	24	9	abeja<16> -> cera<1880> -> abeja<16>
2	21	35	abertura<17> -> grieta<4954> -> abertura<17>
1	44	26	abierto<19> -> abierto<19>
2	13	1	abnegación<21> -> sacrificio<8906> -> abnegación<21>
3	14	7	abogado<22> -> media<6429> -> defensa<2882> -> abogado<22>
18	8	5	abrigar<39> -> mal<6233> -> moral<6691> -> frente<4694> -> tropa<9914> -> componer<2258> -> ordenar<7133> -> fin<4549> -> consumación<2456> -> cumplimiento<2795> -> obsequio<6998> -> agasajo<301> -> agasajar<300> -> halagar<5029> -> alegría<419> -> andaluz<589> -> caracterizado<1692> -> vestido<10183> -> abrigar<39>
2	58	30	abrir<41> -> inaugurar<5377> -> abrir<41>
2	5	6	abrochar<42> -> botón<1393> -> abrochar<42>
1	10	1	abscisa<46> -> abscisa<46>
1	23	8	absoluto<47> -> absoluto<47>
4	20	4	abstraer<58> -> relacionar<8537> -> elemento<3614> -> noción<6905> -> abstraer<58>
1	8	8	aburrido<70> -> aburrido<70>

¹⁶ Resultados obtenidos corriendo el programa vis teniendo como únicos argumentos los archivos de entrada y salida (LC es la longitud del ciclo, FS es el número de flechas salientes, FE- es el número de flechas entrantes, y por último uno de los ciclos más cortos).

<i>LC</i>	<i>FS</i>	<i>FE</i>	<i>Ciclo</i>
1	23	14	acabar<77> -> acabar<77>
1	24	3	acacia<79> -> acacia<79>
4	17	7	académico<81> -> academia<80> -> clase<2046> -> asignatura<961> -> académico<81>
1	21	2	acanto<88> -> acanto<88>
6	8	3	acarrear<90> -> carro<1745> -> enganchar<3793> -> toro<9756> -> adulto<250> -> conllevar<2394> -> acarrear<90>
9	28	12	accidente<97> -> paisaje<7231> -> diferenciar<3311> -> variable<10054> -> pronombre<8057> -> puro<8201> -> construcción<2447> -> madera<6204> -> sustancia<9447> -> accidente<97>
2	38	1119	acción<98> -> ejercicio<3587> -> acción<98>
2	32	21	aceite<102> -> bacalao<1158> -> aceite<102>
2	21	1	acelerador<106> -> acelerar<107> -> acelerador<106>
7	7	4	acentuación<110> -> acento<109> -> articulario<903> -> nasal<6826> -> letra<6024> -> cantar<1651> -> breve<1414> -> acentuación<110>
1	15	3	acepción<112> -> acepción<112>
2	6	3	aceptable<113> -> pasable<7338> -> aceptable<113>
2	5	4	acertado<120> -> acierto<131> -> acertado<120>
2	14	7	acertar<121> -> acierto<131> -> acertar<121>
2	20	25	ácido<130> -> agrio<327> -> ácido<130>
2	7	2	acogida<139> -> hospitalidad<5207> -> acogida<139>
2	24	11	acomodar<143> -> ajustar<371> -> acomodar<143>
2	9	43	acompañar<146> -> acompañamiento<144> -> acompañar<146>
2	6	10	aconsejar<149> -> advertir<256> -> aconsejar<149>
6	7	6	acontecer<150> -> plural<7732> -> pronombre<8057> -> contexto<2481> -> comunicación<2285> -> desear<3099> -> acontecer<150>
2	26	5	acoplar<153> -> ajustar<371> -> acoplar<153>
2	25	3	acotar<161> -> acotación<160> -> acotar<161>
10	3	3	acrecentar<162> -> aumentar<1074> -> prosperar<8094> -> prosperidad<8095> -> suerte<9381> -> concatenación<2289> -> relacionar<8537> -> fenómeno<4489> -> extraordinario<4381> -> añadir<572> -> acrecentar<162>
2	21	19	acreditar<163> -> abonar<26> -> acreditar<163>
6	10	49	actitud<169> -> reflejar<8470> -> cambiar<1594> -> rumbo<8877> -> desprendimiento<3210> -> generosidad<4835> -> actitud<169>
3	12	108	actividad<171> -> tarea<9529> -> misión<6603> -> actividad<171>
1	22	28	activo<172> -> activo<172>
3	13	30	actor<174> -> protagonista<8098> -> personaje<7554> -> actor<174>
2	17	98	actuar<179> -> ejercer<3586> -> actuar<179>
6	8	11	acumular<185> -> progresivo<8037> -> creciente<2671> ->

<i>LC</i>	<i>FS</i>	<i>FE</i>	<i>Ciclo</i>
2	16	9	marea<6348> -> combinar<2182> -> juntar<5884> -> acumular<185>
10	22	20	acusar<190> -> acusado<189> -> acusar<190> adaptar<196> -> aplicado<729> -> esforzar<4042> -> moral<6691> - > frente<4694> -> tropa<9914> -> componer<2258> -> corregir<2601> -> atemperar<1013> -> adecuar<198> -> adaptar<196>
2	25	12	adelantar<200> -> progresar<8034> -> adelantar<200>
3	9	3	adicto<216> -> adepto<207> -> partidario<7331> -> adicto<216>
2	4	2	adinerado<218> -> rico<8774> -> adinerado<218>
2	5	2	adivinanza<219> -> acertijo<122> -> adivinanza<219>
4	18	7	adivinar<220> -> agüero<341> -> presagio<7917> -> vaticinar<10076> -> adivinar<220>
2	17	25	adjetivo<221> -> calificar<1564> -> adjetivo<221>
9	6	22	administración<224> -> administrar<226> -> ingreso<5571> -> ingresar<5570> -> meter<6536> -> aceptar<115> -> pago<7229> -> distrito<3436> -> administrativo<227> -> administración<224>
2	13	16	admiración<229> -> admirable<228> -> admiración<229>
2	6	43	admitir<231> -> reconocer<8411> -> admitir<231>
2	7	2	adobar<232> -> adobo<233> -> adobar<232>
7	14	21	adoptar<234> -> seguir<9034> -> después<3215> -> situación<9200> -> comunicación<2285> -> trato<9858> -> tratamiento<9856> -> adoptar<234>
1	12	1	adoquín<235> -> adoquín<235>
6	5	3	adoración<236> -> apasionado<701> -> entusiasta<3891> -> partidario<7331> -> guerrillero<4993> -> servir<9121> -> adoración<236>
2	13	1	adormidera<239> -> opio<7103> -> adormidera<239>
2	19	16	adverbio<251> -> invariable<5762> -> adverbio<251>
2	13	1	aerolito<260> -> meteorito<6533> -> aerolito<260>
7	8	13	afectado<271> -> padecer<7217> -> resistir<8671> -> deseo<3117> - > conseguir<2415> -> poseer<7808> -> suficiente<9384> -> afectado<271>
4	28	6	afectar<272> -> perjudicar<7525> -> moral<6691> -> interno<5717> -> afectar<272>
3	9	13	afectivo<273> -> emocionar<3674> -> emoción<3671> -> afectivo<273>
2	22	48	afecto<274> -> cariño<1723> -> afecto<274>
2	9	3	afeitar<276> -> rasurar<8302> -> afeitar<276>
9	8	2	afianzar<278> -> confirmar<2366> -> corroborar<2615> -> ratificar<8305> -> anterior<652> -> consonante<2435> -> acentuar<111> -> enfatizar<3779> -> hincapié<5126> -> afianzar<278>
2	10	18	afición<279> -> inclinación<5400> -> afición<279>

<i>LC</i>	<i>FS</i>	<i>FE</i>	<i>Ciclo</i>
9	2	3	afiliado<283> -> asociación<971> -> figura<4529> -> personaje<7554> -> ficción<4515> -> imaginar<5294> -> imaginación<5293> -> real<8326> -> partidario<7331> -> afiliado<283>
2	6	18	afirmar<288> -> asegurar<937> -> afirmar<288>
2	4	13	aflicción<290> -> tristeza<9902> -> aflicción<290>
11	17	3	aflojar<292> -> tirantez<9703> -> tirante<9702> -> falda<4412> -> aguja<344> -> torre<9759> -> minero<6584> -> mina<6581> -> explosivo<4346> -> distensión<3423> -> relajar<8540> -> aflojar<292>
2	6	7	ágil<304> -> ligero<6070> -> ágil<304>
2	15	15	agitar<305> -> alborotar<394> -> agitar<305>
3	7	45	agradable<309> -> agradar<310> -> grato<4941> -> agradable<309>
2	6	2	agravio<316> -> ofensa<7051> -> agravio<316>
2	11	2	agregar<319> -> añadir<572> -> agregar<319>
8	6	7	agricultura<324> -> obtener<7016> -> sustancia<9447> -> sujeto<9392> -> proceso<8006> -> civil<2033> -> ciudad<2030> -> agrario<313> -> agricultura<324>
1	9	1	agrimonia<326> -> agrimonia<326>
2	7	13	agrupación<328> -> agrupar<329> -> agrupación<328>
1	49	172	agua<330> -> agua<330>
2	13	1	aguamarina<331> -> berilo<1294> -> aguamarina<331>
2	15	5	aguantar<333> -> resistir<8671> -> aguantar<333>
2	36	38	agudo<340> -> perspicaz<7562> -> agudo<340>
3	6	25	agujero<346> -> perforar<7509> -> agujerear<345> -> agujero<346>
2	8	1	ahíto<349> -> harto<5039> -> ahíto<349>
2	7	3	ahora<353> -> enseguida<3845> -> ahora<353>
2	29	93	aire<359> -> argón<826> -> aire<359>
8	5	22	aislado<361> -> solitario<9267> -> soledad<9256> -> tonada<9726> -> canción<1624> -> lírico<6107> -> dramática<3512> -> monólogo<6669> -> aislado<361>
3	14	2	ajar<363> -> vejez<10095> -> viejo<10211> -> ajar<363>
2	9	42	ajeno<365> -> impropio<5354> -> ajeno<365>
1	32	5	ajo<368> -> ajo<368>
17	3	1	ajonjolí<369> -> planta<7694> -> célula<1849> -> estructural<4226> -> estructura<4224> -> distribución<3433> -> reparto<8600> -> repartir<8599> -> destino<3228> -> fin<4549> -> consumación<2456> -> cumplimiento<2795> -> obsequio<6998> -> agasajo<301> -> agasajar<300> -> halagar<5029> -> alegría<419> -> ajonjolí<369>
4	7	21	ajustado<370> -> margen<6352> -> real<8326> -> racional<8254> -> ajustado<370>
2	3	10	alabar<376> -> elogiar<3624> -> alabar<376>

<i>LC</i>	<i>FS</i>	<i>FE</i>	<i>Ciclo</i>
1	10	11	alambre<380> -> alambre<380>
2	12	2	alarma<384> -> rebato<8340> -> alarma<384>
8	9	8	albañilería<387> -> ladrillo<5925> -> arcilla<810> -> variable<10054> -> letra<6024> -> cantar<1651> -> componer<2258> -> plana<7683> -> albañilería<387>
1	14	5	albedrío<389> -> albedrío<389>
3	10	6	albergar<390> -> cobijo<2088> -> cobijar<2087> -> albergar<390>
2	9	15	alboroto<395> -> inquietud<5619> -> alboroto<395>
1	9	3	alcalde<397> -> alcalde<397>
12	6	4	alcalino<399> -> álcali<398> -> lejía<6007> -> hidróxido<5106> -> radical<8262> -> tajante<9485> -> brusco<1446> -> estriado<4217> - > estría<4216> -> raya<8309> -> cocaína<2095> -> alcaloide<400> - > alcalino<399>
7	19	7	alcance<401> -> talento<9492> -> dote<3509> -> sobresaliente<9219> -> particular<7328> -> privativo<7989> -> elemento<3614> -> alcance<401>
2	7	5	alcantarilla<402> -> sumidero<9400> -> alcantarilla<402>
2	37	13	alcohol<407> -> etílico<4243> -> alcohol<407>
2	8	13	alcohólico<408> -> bebida<1273> -> alcohólico<408>
12	16	3	aldaba<410> -> argolla<825> -> aro<854> -> flecha<4578> -> torre<9759> -> antena<647> -> oreja<7137> -> sillón<9162> -> cómoda<2215> -> cajón<1531> -> estante<4161> -> balda<1179> -> aldaba<410>
12	14	8	aleación<414> -> elemento<3614> -> cuatro<2749> -> figura<4529> -> procedimiento<8002> -> administrativo<227> -> rama<8273> -> planta<7694> -> célula<1849> -> estructural<4226> -> estructura<4224> -> acero<119> -> aleación<414>
6	18	8	alegar<416> -> citar<2027> -> confirmar<2366> -> firmeza<4564> -> seguir<9034> -> deducir<2877> -> alegar<416>
2	16	11	alegre<418> -> divertido<3444> -> alegre<418>
2	31	7	aleta<424> -> guardabarros<4976> -> aleta<424>
2	7	4	alfabeto<426> -> morse<6709> -> alfabeto<426>
1	27	4	alfiler<429> -> alfiler<429>
2	21	1	algarroba<431> -> algarrobo<432> -> algarroba<431>
4	1	1103	algo<436> -> poco<7741> -> menos<6480> -> cualitativo<2740> -> algo<436>
2	19	19	alimentar<443> -> mantener<6314> -> alimentar<443>
2	14	84	alimento<446> -> alimentación<442> -> alimento<446>
2	8	1	alineación<447> -> alinear<448> -> alineación<447>
6	12	10	alisar<450> -> peinar<7427> -> rastrear<8299> -> rastra<8298> -> allanar<456> -> llana<6129> -> alisar<450>
2	16	6	aliviar<453> -> aligerar<441> -> aliviar<453>
3	27	12	alma<458> -> sentimiento<9093> -> ánimo<623> -> alma<458>

<i>LC</i>	<i>FS</i>	<i>FE</i>	<i>Ciclo</i>
1	17	3	almeja<462> -> almeja<462>
2	17	2	almendro<464> -> almendra<463> -> almendro<464>
2	11	3	almohada<467> -> almohadón<469> -> almohada<467>
2	4	1	almorzar<470> -> almuerzo<472> -> almorzar<470>
2	5	1	almuédano<471> -> alminar<466> -> almuédano<471>
2	12	10	alojar<475> -> alojamiento<474> -> alojar<475>
3	8	44	alrededor<481> -> cerca<1882> -> proximidad<8126> -> alrededor<481>
8	25	24	alta<482> -> formulario<4640> -> fórmula<4638> -> cilindrada<1989> -> cúbico<2752> -> cubo<2755> -> asa<915> -> maleta<6243> -> alta<482>
3	7	1	altanero<483> -> altivo<495> -> arrogante<887> -> altanero<483>
2	11	13	altar<484> -> ara<784> -> altar<484>
2	12	23	alteración<486> -> desorden<3172> -> alteración<486>
2	11	26	alterar<487> -> perturbar<7572> -> alterar<487>
2	7	1	altisonante<493> -> ampuloso<568> -> altisonante<493>
2	57	52	alto<496> -> arriba<881> -> alto<496>
2	21	44	altura<498> -> altitud<494> -> altura<498>
2	15	3	aluminio<506> -> alúmina<505> -> aluminio<506>
6	9	18	alumno<507> -> centro<1876> -> pase<7350> -> toro<9756> -> mamífero<6266> -> clase<2046> -> alumno<507>
2	2	1	aluvial<508> -> aluvión<509> -> aluvial<508>
3	26	4	alzar<513> -> levantar<6032> -> sublevar<9344> -> alzar<513>
1	5	3	ama<514> -> ama<514>
11	6	2	amabilidad<515> -> calidad<1559> -> moral<6691> -> frente<4694> -> tropa<9914> -> componer<2258> -> corregir<2601> -> revisar<8757> -> detenido<3255> -> minucioso<6590> -> detallista<3250> -> amabilidad<515>
8	4	8	amable<516> -> complaciente<2247> -> acceder<92> -> ceder<1831> -> cesar<1916> -> interrumpir<5727> -> seguir<9034> -> atento<1017> -> amable<516>
2	7	2	amaestrar<518> -> adiestrar<217> -> amaestrar<518>
2	9	1	amansar<520> -> domesticar<3482> -> amansar<520>
2	2	2	amargor<524> -> amargo<523> -> amargor<524>
3	27	22	ambiente<532> -> estrato<4207> -> elemento<3614> -> ambiente<532>
2	12	9	ámbito<533> -> perímetro<7518> -> ámbito<533>
2	8	3	aminoácido<542> -> proteína<8104> -> aminoácido<542>
2	13	3	amo<546> -> dueño<3523> -> amo<546>
2	15	1	amonio<549> -> amoniaco<548> -> amonio<549>
2	20	27	amor<551> -> cariño<1723> -> amor<551>
11	6	16	amoroso<555> -> apacible<680> -> pacífico<7214> -> violentar<10236> -> tergiversar<9628> -> confundir<2376> ->

LC FS FE Ciclo

			humillar<5239> -> orgullo<7144> -> desprecio<3207> -> aprecio<746> -> afectuoso<275> -> amoroso<555>
2	6	5	amparar<559> -> proteger<8101> -> amparar<559>
2	6	9	amparo<560> -> defensa<2882> -> amparo<560>
2	4	3	añadido<570> -> postizo<7826> -> añadido<570>
2	6	37	ancho<585> -> holgado<5155> -> ancho<585>
2	16	13	anchura<586> -> amplitud<566> -> anchura<586>
2	21	32	andar<590> -> recorrer<8421> -> andar<590>
2	6	1	anegar<593> -> inundar<5757> -> anegar<593>
2	5	2	anestesiarse<598> -> anestesia<597> -> anestesiarse<598>
2	23	7	ángel<602> -> angelical<604> -> ángel<602>
1	27	1	angélica<603> -> angélica<603>
1	15	3	anglicanismo<606> -> anglicanismo<606>
2	10	9	angustia<612> -> ansiedad<638> -> angustia<612>
3	32	314	animal<621> -> frente<4694> -> temperatura<9587> -> animal<621>
3	26	9	animar<622> -> incitar<5399> -> estimular<4189> -> animar<622>
2	5	2	anión<624> -> ánodo<631> -> anión<624>
2	11	1	aniversario<626> -> cumpleaños<2793> -> aniversario<626>
14	8	3	ano<627> -> excremento<4300> -> eliminar<3619> -> matar<6388> -> yeso<10330> -> sulfato<9393> -> sulfúrico<9394> -> detergente<3257> -> jabonar<5824> -> barba<1219> -> pluma<7729> -> barbilla<1221> -> apéndice<712> -> intestino<5737> -> ano<627>
2	13	58	año<628> -> diciembre<3294> -> año<628>
1	8	4	anochecer<629> -> anochecer<629>
11	4	14	anotar<636> -> registrar<8508> -> reconocer<8411> -> aceptar<115> -> pago<7229> -> campo<1611> -> cuadro<2731> -> cuadrado<2725> -> resultado<8710> -> puntuación<8190> -> puntuar<8193> -> anotar<636>
2	9	5	ansia<637> -> náusea<6834> -> ansia<637>
6	8	49	ante<641> -> indio<5469> -> fusible<4756> -> pase<7350> -> muleta<6760> -> bastón<1253> -> ante<641>
10	20	9	antecedente<643> -> referencia<8463> -> informe<5550> -> radio<8264> -> centro<1876> -> campo<1611> -> laborable<5908> -> festivo<4507> -> señalado<9067> -> sospechoso<9324> -> antecedente<643>
2	6	2	antecesor<645> -> predecesor<7868> -> antecesor<645>
2	1	3	antelación<646> -> anticipación<656> -> antelación<646>
2	5	5	anterioridad<653> -> precedencia<7851> -> anterioridad<653>
2	12	2	antibiótico<655> -> penicilina<7460> -> antibiótico<655>
3	2	2	anticipo<658> -> anticipación<656> -> adelanto<202> -> anticipo<658>
12	13	8	antigüedad<663> -> pasado<7340> -> retener<8719> -> pago<7229> -> campo<1611> -> distancia<3418> -> media<6429> ->

LC FS FE Ciclo

			defensa<2882> -> acusado<189> -> proceso<8006> -> dictar<3299> -> escriba<4012> -> antigüedad<663>
2	18	124	antiguo<664> -> viejo<10211> -> antiguo<664>
7	28	14	anular<673> -> mano<6307> -> jugador<5872> -> deportivo<2973> -> deportividad<2972> -> correcto<2597> -> error<3955> -> anular<673>
2	6	8	anuncio<675> -> anunciar<674> -> anuncio<675>
7	10	5	anverso<676> -> molde<6642> -> pastoso<7370> -> campo<1611> -> blasón<1331> -> ciudad<2030> -> frente<4694> -> anverso<676>
4	13	4	apagar<683> -> desconectar<3068> -> eliminar<3619> -> matar<6388> -> apagar<683>
3	28	105	aparato<684> -> administrativo<227> -> interno<5717> -> aparato<684>
3	22	7	aparejo<692> -> sillar<9160> -> albarda<388> -> aparejo<692>
5	10	17	aparente<694> -> bonito<1362> -> marino<6359> -> mar<6332> -> convencional<2527> -> aparente<694>
2	7	30	apariencia<696> -> aspecto<984> -> apariencia<696>
2	13	23	apartar<699> -> alejar<420> -> apartar<699>
2	14	2	aparte<700> -> párrafo<7317> -> aparte<700>
2	8	2	apelar<707> -> recurrir<8441> -> apelar<707>
2	9	4	apellido<709> -> sobrenombre<9215> -> apellido<709>
2	5	4	apetecer<715> -> desear<3099> -> apetecer<715>
2	9	7	apetito<717> -> gana<4790> -> apetito<717>
7	11	9	aplastar<723> -> abrumar<44> -> agobiar<307> -> abatimiento<11> -> desilusión<3144> -> ilusión<5286> -> quimera<8235> -> aplastar<723>
7	17	19	aplicación<728> -> asociar<973> -> juntar<5884> -> unir<9967> -> acuerdo<183> -> estamento<4151> -> profesional<8025> -> aplicación<728>
2	13	37	apoyar<742> -> basar<1245> -> apoyar<742>
10	5	17	apoyo<743> -> protección<8099> -> enemigo<3769> -> perjudicar<7525> -> moral<6691> -> frente<4694> -> extender<4363> -> espeso<4095> -> macizo<6201> -> vano<10049> -> apoyo<743>
12	8	15	aprender<750> -> experiencia<4329> -> adquirir<243> -> adueñarse<248> -> dominante<3485> -> clase<2046> -> calidad<1559> -> moral<6691> -> frente<4694> -> tropa<9914> -> componer<2258> -> plana<7683> -> aprender<750>
5	9	5	apresar<753> -> garra<4806> -> león<6017> -> carnívoro<1732> -> capturar<1681> -> apresar<753>
2	7	1	apresto<754> -> almidón<465> -> apresto<754>
2	11	9	apretado<756> -> prieto<7963> -> apretado<756>
2	30	19	apretar<757> -> comprimir<2275> -> apretar<757>

<i>LC</i>	<i>FS</i>	<i>FE</i>	<i>Ciclo</i>
5	13	12	aprobar<763> -> calificar<1564> -> calidad<1559> -> moral<6691> -> aprobación<761> -> aprobar<763>
2	3	9	aproximar<770> -> acercar<118> -> aproximar<770>
2	6	15	apto<772> -> aptitud<771> -> apto<772>
2	38	11	apuntar<778> -> apuntador<776> -> apuntar<778>
1	12	5	árabe<785> -> árabe<785>
1	8	10	arado<788> -> arado<788>
1	12	1	aralia<789> -> aralia<789>
2	29	7	araña<790> -> arácnido<787> -> araña<790>
3	12	1	arbitrariedad<795> -> injusto<5588> -> arbitrario<796> -> arbitrariedad<795>
2	18	91	árbol<799> -> planta<7694> -> árbol<799>
11	6	8	ardiente<815> -> ardor<816> -> intrepidez<5746> -> peligro<7434> -> mal<6233> -> moral<6691> -> frente<4694> -> extender<4363> -> espeso<4095> -> pesado<7580> -> caluroso<1583> -> ardiente<815>
2	5	1	arete<822> -> aro<854> -> arete<822>
1	26	1	argonauta<827> -> argonauta<827>
3	12	21	argumento<831> -> refutar<8488> -> argüir<828> -> argumento<831>
5	15	4	arista<835> -> problema<7997> -> aclarar<135> -> aguzar<347> -> filo<4543> -> arista<835>
2	19	108	arma<839> -> defensa<2882> -> arma<839>
2	15	5	armada<840> -> guerra<4989> -> armada<840>
2	30	2	armadillo<841> -> desdentado<3094> -> armadillo<841>
4	13	15	armado<842> -> vestido<10183> -> servir<9121> -> concreto<2320> -> armado<842>
3	12	11	armadura<844> -> montura<6686> -> lente<6014> -> armadura<844>
3	6	7	armario<846> -> cajón<1531> -> estante<4161> -> armario<846>
2	9	29	armazón<847> -> entramado<3873> -> armazón<847>
1	6	1	armenio<848> -> armenio<848>
2	24	17	armonía<849> -> acorde<155> -> armonía<849>
2	5	16	arquitectura<860> -> estructura<4224> -> arquitectura<860>
7	24	13	arrancar<864> -> brusco<1446> -> estriado<4217> -> estría<4216> -> fuste<4759> -> capitel<1673> -> arranque<865> -> arrancar<864>
8	21	11	arrastrar<868> -> penoso<7467> -> padecer<7217> -> mal<6233> -> masculino<6377> -> planta<7694> -> célula<1849> -> envolvente<3913> -> arrastrar<868>
2	7	14	arreglar<872> -> componer<2258> -> arreglar<872>
11	4	2	arrendatario<876> -> arrendamiento<874> -> arrendar<875> -> acostumar<159> -> hábito<5017> -> repetición<8605> -> propósito<8082> -> deseo<3117> -> disfrutar<3388> -> conveniencia<2529> -> renta<8591> -> arrendatario<876>
2	13	2	arresto<880> -> detención<3253> -> arresto<880>

<i>LC</i>	<i>FS</i>	<i>FE</i>	<i>Ciclo</i>
2	3	6	arriesgar<884> -> exponer<4350> -> arriesgar<884>
2	13	17	arrojar<888> -> lanzar<5950> -> arrojar<888>
2	7	5	arruinar<893> -> destruir<3234> -> arruinar<893>
2	17	2	artesano<898> -> artesanía<897> -> artesano<898>
2	5	2	ártico<899> -> norte<6926> -> ártico<899>
2	22	36	articulación<900> -> articular<902> -> articulación<900>
4	19	20	articulado<901> -> artículo<904> -> extensión<4365> -> miembro<6562> -> articulado<901>
8	17	14	artificio<906> -> doblez<3459> -> queda<8213> -> noche<6904> -> salida<8926> -> ocurrencia<7043> -> ingenioso<5561> -> ingenio<5560> -> artificio<906>
2	14	12	artrópodo<913> -> filo<4543> -> artrópodo<913>
2	17	20	asamblea<920> -> deliberante<2921> -> asamblea<920>
2	8	6	ascender<924> -> subir<9339> -> ascender<924>
2	5	1	ascensor<928> -> montacargas<6675> -> ascensor<928>
2	3	1	ascético<930> -> asceta<929> -> ascético<930>
1	7	1	asesino<947> -> asesino<947>
4	2	3	asiduo<957> -> frecuente<4689> -> usual<9993> -> habitual<5018> -> asiduo<957>
9	16	31	asiento<958> -> contable<2461> -> contar<2468> -> merecer<6504> -> corresponder<2608> -> favor<4459> -> empujar<3701> -> puesto<8157> -> ambulante<534> -> asiento<958>
2	5	2	asilo<962> -> refugio<8486> -> asilo<962>
2	26	3	asimilar<964> -> asimilación<963> -> asimilar<964>
2	6	5	asir<966> -> agarrar<299> -> asir<966>
5	18	13	asistir<969> -> interino<5707> -> puesto<8157> -> vestido<10183> -> servir<9121> -> asistir<969>
2	8	1	asolar<975> -> arrasar<866> -> asolar<975>
2	4	2	asomo<980> -> indicio<5462> -> asomo<980>
1	9	2	aspar<982> -> aspar<982>
8	3	6	aspereza<985> -> calidad<1559> -> moral<6691> -> frente<4694> -> fachada<4399> -> especial<4075> -> diferente<3312> -> desigual<3142> -> aspereza<985>
19	17	6	aspiración<987> -> pretensión<7951> -> petición<7596> -> favor<4459> -> concesión<2302> -> ceder<1831> -> cesar<1916> -> emperador<3690> -> vasallo<10066> -> súbdito<9335> -> sujeto<9392> -> padecer<7217> -> incomodidad<5409> -> disgusto<3391> -> tristeza<9902> -> triste<9901> -> pesadumbre<7581> -> desazón<3042> -> inquietud<5619> -> aspiración<987>
8	16	13	aspirar<989> -> superficie<9409> -> apreciar<745> -> precio<7856> -> conseguir<2415> -> poseer<7808> -> impulso<5364> -> inspirar<5651> -> aspirar<989>

<i>LC</i>	<i>FS</i>	<i>FE</i>	<i>Ciclo</i>
2	6	20	astuto<999> -> diestro<3305> -> astuto<999>
2	10	133	asunto<1001> -> cuestión<2774> -> asunto<1001>
2	13	18	atacar<1003> -> acometer<140> -> atacar<1003>
1	3	7	atado<1004> -> atado<1004>
2	6	6	atadura<1005> -> ligadura<6066> -> atadura<1005>
3	4	19	atar<1009> -> ligadura<6066> -> unir<9967> -> atar<1009>
2	11	2	atasco<1010> -> obstrucción<7013> -> atasco<1010>
3	12	20	atender<1015> -> concentrar<2297> -> atención<1014> -> atender<1015>
1	2	1	ático<1023> -> ático<1023>
2	16	2	atletismo<1031> -> lanzamiento<5949> -> atletismo<1031>
2	17	34	atraer<1041> -> adhesión<212> -> atraer<1041>
7	4	4	atreverse<1045> -> faltar<4424> -> morir<6706> -> llama<6124> -> lama<5931> -> musgo<6792> -> crecer<2669> -> atreverse<1045>
5	5	13	atribuir<1049> -> asignar<960> -> nombrar<6913> -> honorífico<5176> -> conferir<2353> -> atribuir<1049>
2	3	2	audacia<1060> -> osadía<7177> -> audacia<1060>
2	6	4	audición<1062> -> concierto<2306> -> audición<1062>
7	16	3	audiencia<1063> -> escuchar<4026> -> aviso<1132> -> indicio<5462> -> clase<2046> -> aula<1071> -> sala<8916> -> audiencia<1063>
2	9	1	auditor<1066> -> auditoría<1067> -> auditor<1066>
7	8	2	auditorio<1068> -> oyente<7207> -> escucha<4025> -> radio<8264> -> centro<1876> -> asociación<971> -> retórica<8730> -> auditorio<1068>
2	6	20	aumento<1075> -> ascenso<927> -> aumento<1075>
2	30	1	aurícula<1078> -> ventrículo<10135> -> aurícula<1078>
6	13	21	ausencia<1080> -> desconocer<3072> -> cambiar<1594> -> rumbo<8877> -> seguir<9034> -> detrás<3264> -> ausencia<1080>
9	10	6	ausente<1082> -> abstraído<59> -> absorto<52> -> queda<8213> -> campana<1605> -> sonar<9289> -> ruido<8872> -> comentario<2194> -> murmuración<6782> -> ausente<1082>
5	6	1	auténtica<1086> -> certificación<1910> -> certificar<1912> -> registrar<8508> -> reconocer<8411> -> auténtica<1086>
2	14	2	autómata<1092> -> robot<8807> -> autómeta<1092>
2	27	5	autonomía<1095> -> independencia<5454> -> autonomía<1095>
7	11	30	autor<1098> -> delito<2932> -> culpa<2783> -> consciente<2410> -> pleno<7723> -> centro<1876> -> factor<4403> -> autor<1098>
3	17	79	autoridad<1099> -> establecer<4136> -> residencia<8663> -> autoridad<1099>
6	7	8	autoritario<1100> -> abusar<73> -> experiencia<4329> -> adquirir<243> -> adueñarse<248> -> dominante<3485> -> autoritario<1100>

<i>LC</i>	<i>FS</i>	<i>FE</i>	<i>Ciclo</i>
11	7	9	autorizado<1102> -> digno<3329> -> corresponder<2608> -> favor<4459> -> empujar<3701> -> puesto<8157> -> camuflar<1612> -> engañar<3792> -> realidad<8329> -> real<8326> -> auténtico<1087> -> autorizado<1102>
2	4	4	avalar<1107> -> garantizar<4799> -> avalar<1107>
12	11	6	avanzado<1110> -> novedad<6939> -> salir<8929> -> grieta<4954> -> doloroso<3478> -> compasión<2230> -> ternura<9638> -> tierno<9674> -> reciente<8380> -> acabado<76> -> avejentar<1117> -> viejo<10211> -> avanzado<1110>
2	6	9	avanzar<1111> -> adelante<201> -> avanzar<1111>
3	7	2	avaro<1114> -> miserable<6599> -> avariento<1113> -> avaro<1114>
2	25	93	ave<1116> -> ala<374> -> ave<1116>
2	10	2	avellana<1118> -> avellano<1119> -> avellana<1118>
2	3	14	averiguar<1126> -> indagar<5443> -> averiguar<1126>
2	33	32	avión<1130> -> aeroplano<262> -> avión<1130>
13	4	17	ayudar<1140> -> fin<4549> -> consumación<2456> -> cumplimiento<2795> -> obsequio<6998> -> agasajo<301> -> festejo<4504> -> festejar<4503> -> divertir<3445> -> recrear<8426> -> crear<2666> -> vida<10206> -> mantener<6314> -> ayudar<1140>
2	12	3	ayuno<1142> -> ayunar<1141> -> ayuno<1142>
2	7	1	azada<1144> -> pala<7234> -> azada<1144>
2	2	1	azotaina<1148> -> zurra<10356> -> azotaina<1148>
12	18	5	azote<1150> -> mano<6307> -> inclusive<5403> -> aplicar<730> -> fin<4549> -> consumación<2456> -> cumplimiento<2795> -> obsequio<6998> -> agasajo<301> -> festejo<4504> -> festejar<4503> -> azotar<1149> -> azote<1150>
1	11	1	azteca<1151> -> azteca<1151>
2	22	30	azúcar<1152> -> glucosa<4879> -> azúcar<1152>
2	19	4	azufre<1155> -> sulfuroso<9395> -> azufre<1155>
1	10	2	bacilo<1159> -> bacilo<1159>
2	4	1	báculo<1161> -> cayado<1823> -> báculo<1161>
2	5	3	badajo<1162> -> campana<1605> -> badajo<1162>
2	14	19	baile<1166> -> bailarín<1165> -> baile<1166>
2	18	15	bajar<1169> -> apear<705> -> bajar<1169>
2	7	1	balancear<1174> -> mecer<6424> -> balancear<1174>
2	11	8	balanza<1176> -> pesar<7582> -> balanza<1176>
2	11	22	balón<1186> -> fútbol<4760> -> balón<1186>
7	20	13	baloncesto<1187> -> aro<854> -> espiga<4100> -> caracol<1688> -> manto<6317> -> repliegue<8610> -> doble<3457> -> baloncesto<1187>
2	2	1	banano<1193> -> bananero<1192> -> banano<1193>
3	32	3	banca<1195> -> banquero<1209> -> bancario<1197> -> banca<1195>

<i>LC</i>	<i>FS</i>	<i>FE</i>	<i>Ciclo</i>
7	30	11	banco<1198> -> macizo<6201> -> vano<10049> -> irreal<5799> -> real<8326> -> partidario<7331> -> bando<1205> -> banco<1198>
4	36	18	banda<1199> -> faja<4409> -> centro<1876> -> extremo<4388> -> banda<1199>
2	21	10	bandera<1201> -> asta<990> -> bandera<1201>
2	13	1	banderilla<1202> -> banderillero<1203> -> banderilla<1202>
2	19	7	baño<1208> -> aseo<943> -> baño<1208>
2	12	2	baquelita<1212> -> fenol<4488> -> baquelita<1212>
2	41	26	barra<1229> -> palanca<7240> -> barra<1229>
2	2	1	barracón<1231> -> caseta<1774> -> barracón<1231>
9	9	1	barranco<1232> -> erosión<3951> -> prestigio<7941> -> reputación<8640> -> crédito<2674> -> préstamo<7937> -> convenir<2532> -> pareja<7305> -> barco<1223> -> barranco<1232>
8	28	2	barrera<1236> -> lanzamiento<5949> -> cohete<2124> -> explosivo<4346> -> distensión<3423> -> relajación<8538> -> dominio<3488> -> frontera<4708> -> barrera<1236>
2	8	8	barrio<1238> -> arrabal<861> -> barrio<1238>
3	18	18	barro<1240> -> búcaro<1451> -> botijo<1391> -> barro<1240>
9	7	6	basa<1242> -> estatua<4165> -> esculpir<4032> -> madera<6204> -> sustancia<9447> -> cambiar<1594> -> referencia<8463> -> informe<5550> -> documental<3468> -> basa<1242>
2	76	62	base<1246> -> béisbol<1277> -> base<1246>
2	23	5	bastidor<1250> -> vidriera<10209> -> bastidor<1250>
7	15	10	batalla<1255> -> justa<5893> -> lanza<5946> -> bomba<1356> -> explosivo<4346> -> consonante<2435> -> choque<1958> -> batalla<1255>
2	9	1	batallar<1256> -> pelear<7432> -> batallar<1256>
2	5	1	bate<1258> -> béisbol<1277> -> bate<1258>
1	22	4	batido<1260> -> batido<1260>
3	4	2	bautizo<1264> -> bautizar<1263> -> bautismo<1262> -> bautizo<1264>
2	12	17	beber<1272> -> bebida<1273> -> beber<1272>
2	8	42	beneficio<1289> -> ganancia<4794> -> beneficio<1289>
6	7	3	benéfico<1290> -> beneficencia<1287> -> primario<7967> -> seguir<9034> -> cursar<2810> -> centro<1876> -> benéfico<1290>
2	14	2	betún<1298> -> asfalto<953> -> betún<1298>
7	16	3	biblioteca<1301> -> clasificar<2051> -> puesto<8157> -> camuflar<1612> -> presencia<7921> -> aspecto<984> -> edificio<3556> -> biblioteca<1301>
2	6	9	bicicleta<1302> -> pedal<7411> -> bicicleta<1302>
4	28	151	bien<1303> -> gana<4790> -> deseo<3117> -> disfrutar<3388> -> bien<1303>
2	2	2	biliar<1310> -> bilis<1311> -> biliar<1310>

<i>LC</i>	<i>FS</i>	<i>FE</i>	<i>Ciclo</i>
2	17	13	billete<1313> -> lotería<6172> -> billete<1313>
2	4	2	biografía<1314> -> vida<10206> -> biografía<1314>
2	12	28	biología<1315> -> citología<2028> -> biología<1315>
7	23	91	blanco<1325> -> claro<2044> -> iluminar<5285> -> templo<9592> -> real<8326> -> rey<8768> -> ajedrez<364> -> blanco<1325>
2	11	36	blando<1327> -> tierno<9674> -> blando<1327>
2	21	1	blues<1335> -> rock<8814> -> blues<1335>
2	42	50	boca<1337> -> orificio<7151> -> boca<1337>
2	27	3	bocadillo<1338> -> globo<4875> -> bocadillo<1338>
2	23	2	bodega<1344> -> sótano<9328> -> bodega<1344>
4	37	19	bola<1349> -> cruzado<2720> -> delantero<2910> -> pelota<7438> -> bola<1349>
2	9	2	bono<1363> -> vale<10023> -> bono<1363>
3	8	4	bordado<1367> -> aguja<344> -> bordar<1368> -> bordado<1367>
2	16	32	borde<1369> -> bordo<1372> -> borde<1369>
1	2	1	borgoña<1374> -> borgoña<1374>
2	24	2	borra<1376> -> algodón<437> -> borra<1376>
1	13	16	bóveda<1394> -> bóveda<1394>
2	18	1	bovino<1395> -> vacuno<10014> -> bovino<1395>
2	1	4	boxeador<1396> -> púgil<8158> -> boxeador<1396>
2	13	1	braga<1399> -> pañal<7264> -> braga<1399>
2	4	3	brasa<1401> -> carbón<1696> -> brasa<1401>
2	41	32	brazo<1409> -> bracero<1398> -> brazo<1409>
2	33	2	brea<1410> -> calafatear<1537> -> brea<1410>
2	24	2	brigada<1418> -> subteniente<9356> -> brigada<1418>
2	4	18	brillo<1422> -> lustre<6192> -> brillo<1422>
2	9	2	bronquio<1440> -> tráquea<9842> -> bronquio<1440>
2	17	14	brotar<1441> -> germinar<4854> -> brotar<1441>
11	18	3	bruto<1449> -> tara<9524> -> embalaje<3629> -> cubierta<2753> -> cámara<1591> -> granero<4928> -> grano<4933> -> pústula<8207> -> pus<8205> -> tumor<9935> -> crecimiento<2672> -> bruto<1449>
1	4	97	buen<1455> -> buen<1455>
2	9	5	buey<1456> -> toro<9756> -> buey<1456>
2	15	1	bufón<1458> -> truhán<9923> -> bufón<1458>
2	30	1	búho<1459> -> rapaz<8282> -> búho<1459>
12	5	2	bullicio<1463> -> tumulto<9936> -> motín<6723> -> levantar<6032> -> sonar<9289> -> citar<2027> -> confirmar<2366> -> administrar<226> -> ingreso<5571> -> ingresar<5570> -> meter<6536> -> inquietud<5619> -> bullicio<1463>
9	8	3	bullicioso<1464> -> alborotador<393> -> alborotar<394> -> amotinar<558> -> motín<6723> -> levantar<6032> -> campo<1611> -> laborable<5908> -> festivo<4507> -> bullicioso<1464>
6	11	33	buque<1467> -> cubierta<2753> -> editorial<3559> -> artículo<904>

<i>LC</i>	<i>FS</i>	<i>FE</i>	<i>Ciclo</i>
			-> luna<6187> -> marina<6356> -> buque<1467>
1	6	1	burdeos<1469> -> burdeos<1469>
2	10	10	burla<1471> -> broma<1432> -> burla<1471>
7	8	4	burlar<1472> -> chasquear<1935> -> látigo<5970> -> cincha<1995> -> faja<4409> -> centro<1876> -> pase<7350> -> burlar<1472>
3	2	3	burocrático<1475> -> burocracia<1474> -> papeleo<7276> -> burocrático<1475>
3	13	8	busca<1478> -> correr<2605> -> recorrer<8421> -> busca<1478>
2	5	30	buscar<1479> -> encontrar<3748> -> buscar<1479>
2	19	1	buzón<1483> -> correo<2604> -> buzón<1483>
9	6	3	cabal<1484> -> íntegro<5679> -> cohecho<2120> -> tierra<9675> -> superficie<9409> -> apreciar<745> -> cariño<1723> -> expresión<4355> -> matemático<6391> -> cabal<1484>
2	35	61	caballería<1489> -> cabaleresco<1488> -> caballería<1489>
14	20	8	caballero<1490> -> plaza<7714> -> feria<4491> -> promoción<8053> -> mejor<6453> -> artículo<904> -> luna<6187> -> marina<6356> -> mar<6332> -> salado<8917> -> ameno<538> -> encanto<3719> -> apelativo<708> -> mote<6722> -> caballero<1490>
2	2	1	caballito<1491> -> tiovivo<9690> -> caballito<1491>
3	29	44	caballo<1492> -> burro<1476> -> asno<970> -> caballo<1492>
2	4	22	cabello<1497> -> pelo<7437> -> cabello<1497>
6	27	96	cabeza<1500> -> anterior<652> -> pasada<7339> -> ligero<6070> -> leve<6034> -> pesado<7580> -> cabeza<1500>
2	20	33	cabo<1505> -> tropa<9914> -> cabo<1505>
14	2	1	cabrió<1509> -> cabra<1507> -> cabrilla<1508> -> rebotar<8347> -> despedida<3182> -> canción<1624> -> era<3946> -> llana<6129> -> yeso<10330> -> sulfato<9393> -> sulfúrico<9394> -> detergente<3257> -> jabonar<5824> -> barba<1219> -> cabrió<1509>
2	15	4	cacao<1513> -> chocolate<1956> -> cacao<1513>
3	6	1	cachaza<1516> -> flema<4580> -> calma<1575> -> cachaza<1516>
3	3	1	cachorro<1518> -> cría<2685> -> criar<2688> -> cachorro<1518>
5	25	19	cadena<1520> -> programa<8033> -> radio<8264> -> círculo<2013> -> circuito<2009> -> cadena<1520>
2	8	7	cadera<1522> -> pelvis<7443> -> cadera<1522>
2	44	48	caer<1525> -> incurrir<5441> -> caer<1525>
8	17	10	caída<1527> -> colgadura<2149> -> celebración<1839> -> aclamación<132> -> ovación<7194> -> victoria<10204> -> coche<2099> -> cerdo<1890> -> caída<1527>
2	58	22	caja<1528> -> ataúd<1011> -> caja<1528>
2	22	13	cal<1532> -> argamasa<823> -> cal<1532>
2	22	4	calamar<1538> -> tinta<9687> -> calamar<1538>
15	8	1	calco<1549> -> carbón<1696> -> marrón<6366> -> castaño<1781> -> espinoso<4103> -> espina<4101> -> circo<2008> -> carro<1745> ->

LC FS FE Ciclo

			lanza<5946> -> tubo<9928> -> crema<2679> -> selecto<9041> -> reputación<8640> -> crédito<2674> -> préstamo<7937> -> calco<1549>
2	13	14	calcular<1551> -> evaluar<4256> -> calcular<1551>
7	18	17	cálculo<1552> -> vesícula<10181> -> ampolla<567> -> vidrio<10210> -> fusión<4758> -> economía<3544> -> presupuesto<7948> -> cálculo<1552>
11	12	3	caldo<1554> -> aderezo<209> -> pulsera<8177> -> aro<854> -> flecha<4578> -> afilado<281> -> afilar<282> -> lapicero<5951> -> tubo<9928> -> crema<2679> -> sopa<9297> -> caldo<1554>
2	12	2	calendario<1555> -> almanaque<461> -> calendario<1555>
2	15	4	cáliz<1567> -> flor<4587> -> cáliz<1567>
2	4	6	caliza<1568> -> calcita<1548> -> caliza<1568>
2	6	5	callar<1572> -> omitir<7083> -> callar<1572>
2	26	56	calor<1580> -> ardor<816> -> calor<1580>
2	25	24	calzado<1587> -> descalzo<3047> -> calzado<1587>
2	37	17	cama<1589> -> manta<6312> -> cama<1589>
2	20	69	cambio<1595> -> cambiar<1594> -> cambio<1595>
2	14	1	camilla<1599> -> falda<4412> -> camilla<1599>
6	18	5	campamento<1604> -> barraca<1230> -> caña<1614> -> bota<1386> -> gallo<4783> -> plano<7693> -> campamento<1604>
2	11	3	campanario<1607> -> espadaña<4062> -> campanario<1607>
5	5	2	campesino<1609> -> campo<1611> -> blasón<1331> -> ciudad<2030> -> agrario<313> -> campesino<1609>
2	61	19	canal<1616> -> desagüe<3012> -> canal<1616>
2	8	2	canalizar<1617> -> encauzar<3730> -> canalizar<1617>
2	12	1	canelo<1630> -> canela<1629> -> canelo<1630>
10	6	5	cañería<1631> -> tubería<9927> -> tubo<9928> -> envase<3900> -> artículo<904> -> luna<6187> -> noche<6904> -> salida<8926> -> acometida<141> -> acometer<140> -> cañería<1631>
6	19	1	canilla<1634> -> tejer<9560> -> ganchillo<4796> -> horquilla<5196> -> puntal<8186> -> caña<1614> -> canilla<1634>
2	1	2	cannabis<1639> -> cáñamo<1618> -> cannabis<1639>
2	28	1	canon<1641> -> canónico<1643> -> canon<1641>
2	30	7	cañón<1642> -> pluma<7729> -> cañón<1642>
2	4	6	cansancio<1645> -> fatiga<4454> -> cansancio<1645>
1	7	1	cantábrico<1648> -> cantábrico<1648>
6	38	9	canto<1657> -> corte<2627> -> criar<2688> -> educar<3562> -> intelectual<5681> -> espiritual<4110> -> canto<1657>
2	14	2	caolín<1659> -> porcelana<7785> -> caolín<1659>
4	22	44	capa<1660> -> manga<6291> -> cónico<2386> -> superficie<9409> -> capa<1660>
2	24	88	capacidad<1661> -> aptitud<771> -> capacidad<1661>

<i>LC</i>	<i>FS</i>	<i>FE</i>	<i>Ciclo</i>
2	11	51	capaz<1664> -> dispuesto<3415> -> capaz<1664>
2	11	1	capilaridad<1667> -> capilar<1666> -> capilaridad<1667>
2	28	1	capirote<1670> -> cucurucho<2762> -> capirote<1670>
4	34	12	capital<1671> -> economía<3544> -> presupuesto<7948> -> empresa<3698> -> capital<1671>
2	19	5	capitán<1672> -> comandante<2173> -> capitán<1672>
8	8	5	capricho<1677> -> antojo<668> -> embarazo<3634> -> óvulo<7202> -> partir<7333> -> clase<2046> -> refinamiento<8467> -> gusto<5006> -> capricho<1677>
2	10	5	caprichoso<1678> -> antojo<668> -> caprichoso<1678>
2	15	6	capullo<1684> -> bellota<1283> -> capullo<1684>
2	24	58	cara<1685> -> frente<4694> -> cara<1685>
8	23	1	carabinero<1687> -> crustáceo<2718> -> abdomen<13> -> vientre<10213> -> corresponder<2608> -> favor<4459> -> empujar<3701> -> puesto<8157> -> carabinero<1687>
6	22	89	carácter<1689> -> diferenciar<3311> -> variable<10054> -> letra<6024> -> cantar<1651> -> característico<1691> -> carácter<1689>
3	12	46	característica<1690> -> diferenciar<3311> -> diferencia<3309> -> característica<1690>
3	7	2	carboncillo<1698> -> carbonizar<1700> -> carbón<1696> -> carboncillo<1698>
2	10	17	carbono<1701> -> orgánico<7138> -> carbono<1701>
2	11	3	carburante<1703> -> bencina<1285> -> carburante<1703>
5	21	12	cardinal<1707> -> cáncer<1623> -> constelación<2441> -> aspecto<984> -> orientación<7146> -> cardinal<1707>
5	35	2	cardo<1708> -> espinoso<4103> -> espina<4101> -> circo<2008> -> cima<1992> -> cardo<1708>
2	45	41	carga<1712> -> cargar<1716> -> carga<1712>
3	7	5	cargado<1713> -> espeso<4095> -> pesado<7580> -> cargado<1713>
2	11	123	cargo<1718> -> cuidado<2777> -> cargo<1718>
2	2	13	cariñoso<1724> -> afectuoso<275> -> cariñoso<1724>
4	6	1	caritativo<1725> -> caridad<1721> -> sensibilidad<9080> -> humano<5231> -> caritativo<1725>
9	12	1	carmín<1727> -> afeite<277> -> aderezo<209> -> pulsera<8177> -> aro<854> -> púrpura<8202> -> violáceo<10231> -> lila<6072> -> morado<6690> -> carmín<1727>
2	16	6	carnero<1730> -> oveja<7199> -> carnero<1730>
2	16	1	carpa<1736> -> circo<2008> -> carpa<1736>
2	46	27	carrera<1739> -> carrerilla<1740> -> carrera<1739>
7	4	2	cartesiano<1755> -> cartesianismo<1754> -> derivada<2987> -> cero<1900> -> contar<2468> -> enumerar<3893> -> ordenada<7128> -> cartesiano<1755>

<i>LC</i>	<i>FS</i>	<i>FE</i>	<i>Ciclo</i>
4	17	8	cartón<1758> -> cajetilla<1529> -> paquete<7279> -> envoltorio<3911> -> cartón<1758>
3	14	95	casa<1761> -> edificio<3556> -> vivienda<10280> -> casa<1761>
2	25	3	casar<1764> -> unir<9967> -> casar<1764>
6	7	11	cáscara<1767> -> cubierta<2753> -> editorial<3559> -> artículo<904> -> determinante<3261> -> partir<7333> -> cáscara<1767>
2	39	14	casco<1769> -> herradura<5088> -> casco<1769>
11	4	27	casi<1775> -> cerca<1882> -> proximidad<8126> -> calidad<1559> -> moral<6691> -> frente<4694> -> tropa<9914> -> componer<2258> -> corregir<2601> -> error<3955> -> acto<173> -> casi<1775>
2	14	5	casilla<1776> -> ajedrez<364> -> casilla<1776>
4	20	46	caso<1778> -> problema<7997> -> operar<7099> -> aplicar<730> -> caso<1778>
2	12	3	castaña<1780> -> castaño<1781> -> castaña<1780>
2	15	11	castigar<1786> -> látigo<5970> -> castigar<1786>
2	6	28	castigo<1787> -> corrección<2596> -> castigo<1787>
2	9	1	catalogar<1795> -> registrar<8508> -> catalogar<1795>
2	11	3	cátedra<1801> -> catedrático<1803> -> cátedra<1801>
3	7	3	catedral<1802> -> capítulo<1674> -> cabildo<1502> -> catedral<1802>
3	13	81	categoría<1804> -> diferente<3312> -> modo<6629> -> categoría<1804>
2	2	1	catión<1807> -> ion<5788> -> catión<1807>
1	8	33	católico<1809> -> católico<1809>
2	11	106	causa<1814> -> proceso<8006> -> causa<1814>
2	5	2	cavar<1820> -> profundizar<8030> -> cavar<1820>
11	9	7	cazador<1825> -> instinto<5657> -> automático<1093> -> indefectible<5448> -> faltar<4424> -> ofender<7049> -> demostrar<2947> -> asegurar<937> -> favor<4459> -> empujar<3701> -> puesto<8157> -> cazador<1825>
9	17	12	cazar<1826> -> matar<6388> -> sello<9043> -> estampar<4154> -> pensar<7907> -> prensa<7906> -> englobar<3800> -> abarcar<8> -> caza<1824> -> cazar<1826>
2	19	4	cebolla<1829> -> bulbo<1460> -> cebolla<1829>
2	13	3	cefalópodo<1833> -> pulpo<8172> -> cefalópodo<1833>
5	19	2	ceja<1835> -> pelo<7437> -> raya<8309> -> moral<6691> -> frente<4694> -> ceja<1835>
3	14	1	celda<1837> -> óvulo<7202> -> célula<1849> -> celda<1837>
15	8	4	celdilla<1838> -> nicho<6879> -> cementerio<1853> -> animación<618> -> participación<7322> -> lotería<6172> -> casero<1773> -> remedio<8563> -> atajar<1006> -> rebaño<8337> -> manada<6268> -> salvaje<8950> -> portar<7797> -> vela<10097> ->

<i>LC</i>	<i>FS</i>	<i>FE</i>	<i>Ciclo</i>
8	5	9	cera<1880> -> celdilla<1838> celeste<1843> -> claro<2044> -> discurso<3378> -> extensión<4365> -> conectar<2346> -> funcionamiento<4736> -> puesta<8156> -> astro<994> -> celeste<1843>
2	5	1	cenar<1856> -> cena<1855> -> cenar<1856>
2	18	1	cenizo<1861> -> ceniciento<1857> -> cenizo<1861>
13	11	3	censo<1863> -> pago<7229> -> campo<1611> -> distancia<3418> -> media<6429> -> defensa<2882> -> acusado<189> -> proceso<8006> - > dictar<3299> -> leer<5991> -> intimidad<5738> -> parcela<7297> - > catastro<1798> -> censo<1863>
2	8	9	censurar<1866> -> criticar<2703> -> censurar<1866>
1	6	3	centígrado<1868> -> centígrado<1868>
2	23	37	central<1872> -> centro<1876> -> central<1872>
2	22	2	cepo<1879> -> trampa<9811> -> cepo<1879>
2	4	11	cercano<1885> -> próximo<8127> -> cercano<1885>
2	11	7	cerda<1889> -> cerdo<1890> -> cerda<1889>
3	8	12	cereal<1891> -> cebada<1827> -> grano<4933> -> cereal<1891>
2	8	1	cerebelo<1892> -> encéfalo<3731> -> cerebelo<1892>
2	22	5	cerebro<1894> -> encéfalo<3731> -> cerebro<1894>
2	18	19	ceremonia<1895> -> acto<173> -> ceremonia<1895>
2	16	2	cerezo<1898> -> cereza<1897> -> cerezo<1898>
2	8	1	cerilla<1899> -> fósforo<4654> -> cerilla<1899>
5	3	12	cerrada<1901> -> lomo<6165> -> cerdo<1890> -> oreja<7137> -> apéndice<712> -> cerrada<1901>
6	15	23	cerrado<1902> -> acento<109> -> elemento<3614> -> cuatro<2749> - > figura<4529> -> carta<1750> -> cerrado<1902>
2	68	38	cerrar<1904> -> cicatrizar<1971> -> cerrar<1904>
8	15	4	certamen<1907> -> estimular<4189> -> aumentar<1074> -> prosperar<8094> -> prosperidad<8095> -> bienestar<1306> -> vida<10206> -> mantener<6314> -> certamen<1907>
2	6	5	cerveza<1913> -> cebada<1827> -> cerveza<1913>
3	4	4	cesión<1918> -> favor<4459> -> concesión<2302> -> cesión<1918>
10	17	2	cesta<1920> -> bolsa<1353> -> beca<1275> -> embozo<3650> -> doblez<3459> -> queda<8213> -> dado<2824> -> cúbico<2752> -> cubo<2755> -> asa<915> -> cesta<1920>
2	23	2	cetáceo<1922> -> ballena<1184> -> cetáceo<1922>
5	5	2	chaleco<1924> -> manga<6291> -> bomba<1356> -> tinaja<9684> -> centro<1876> -> chaleco<1924>
9	14	7	chapa<1927> -> botella<1390> -> cuartillo<2746> -> castellano<1783> -> castillo<1788> -> cubierta<2753> -> editorial<3559> -> firmar<4562> -> estampar<4154> -> chapa<1927>
2	8	3	checo<1938> -> bohemio<1348> -> checo<1938>
2	32	1	chinche<1945> -> hemíptero<5064> -> chinche<1945>

<i>LC</i>	<i>FS</i>	<i>FE</i>	<i>Ciclo</i>
6	11	4	chocar<1954> -> desentonar<3113> -> salir<8929> -> suerte<9381> -> concatenación<2289> -> encontrar<3748> -> chocar<1954>
2	8	1	chulería<1963> -> chulo<1964> -> chulería<1963>
2	11	1	cíclico<1972> -> ciclo<1975> -> cíclico<1972>
2	21	1	ciclón<1976> -> huracán<5247> -> ciclón<1976>
2	18	19	cielo<1978> -> azul<1156> -> cielo<1978>
2	14	87	ciencia<1979> -> rama<8273> -> ciencia<1979>
8	8	35	científico<1981> -> relacionar<8537> -> totalidad<9774> -> calidad<1559> -> moral<6691> -> frente<4694> -> tropa<9914> -> componer<2258> -> científico<1981>
1	4	450	cierto<1983> -> cierto<1983>
11	18	7	cifra<1985> -> sello<9043> -> queda<8213> -> noche<6904> -> salida<8926> -> solucionar<9276> -> problema<7997> -> operar<7099> -> combinación<2180> -> prefijar<7887> -> prefijo<7888> -> cifra<1985>
10	6	9	cigarro<1988> -> cigarrillo<1987> -> picadura<7611> -> picado<7609> -> cámara<1591> -> departamento<2965> -> afin<284> -> afinidad<286> -> molécula<6645> -> puro<8201> -> cigarro<1988>
3	17	19	cilindro<1991> -> tubo<9928> -> cilíndrico<1990> -> cilindro<1991>
2	11	2	cinematógrafo<2000> -> cine<1998> -> cinematógrafo<2000>
5	27	18	cinta<2003> -> cinematografía<1999> -> fotografía<4658> -> impresionable<5346> -> impresionar<5348> -> cinta<2003>
2	5	1	cinto<2004> -> cinturón<2006> -> cinto<2004>
3	7	12	cintura<2005> -> costilla<2652> -> espalda<4064> -> cintura<2005>
5	3	1	circense<2007> -> circo<2008> -> acróbata<167> -> acrobacia<166> -> trapecio<9840> -> circense<2007>
8	18	36	circular<2011> -> círculo<2013> -> circuito<2009> -> perímetro<7518> -> geometría<4848> -> trata<9854> -> tráfico<9793> -> circulación<2010> -> circular<2011>
2	6	15	circunferencia<2014> -> centro<1876> -> circunferencia<2014>
3	18	54	circunstancia<2018> -> condicionar<2339> -> condición<2336> -> circunstancia<2018>
6	7	2	circunstancial<2019> -> accidental<96> -> puesto<8157> -> destacamento<3217> -> envío<3909> -> modo<6629> -> circunstancial<2019>
2	10	1	ciruelo<2021> -> ciruela<2020> -> ciruelo<2021>
14	11	6	cirugía<2022> -> curar<2805> -> remedio<8563> -> atajar<1006> -> atajo<1007> -> senda<9071> -> vereda<10155> -> acera<117> -> calle<1573> -> correr<2605> -> lidiar<6062> -> conflictivo<2369> -> problema<7997> -> operar<7099> -> cirugía<2022>
2	3	2	citación<2026> -> comparecer<2225> -> citación<2026>
2	6	3	citoplasma<2029> -> célula<1849> -> citoplasma<2029>

<i>LC</i>	<i>FS</i>	<i>FE</i>	<i>Ciclo</i>
7	18	11	ciudadano<2032> -> vecino<10079> -> obtener<7016> -> sustancia<9447> -> sujeto<9392> -> proceso<8006> -> civil<2033> -> ciudadano<2032>
2	9	1	civilizar<2035> -> civilización<2034> -> civilizar<2035>
4	18	10	clara<2039> -> sustancia<9447> -> discurso<3378> -> pronunciado<8063> -> clara<2039>
2	5	1	claraboya<2040> -> tragaluz<9796> -> claraboya<2040>
4	9	1	claroscuro<2045> -> fotografía<4658> -> oscuro<7184> -> sombrear<9280> -> claroscuro<2045>
3	9	15	clasificación<2049> -> clasificar<2051> -> clase<2046> -> clasificación<2049>
11	17	3	cláusula<2053> -> apartado<697> -> decreto<2870> -> emanar<3628> -> pleno<7723> -> quiniela<8239> -> lotería<6172> -> casero<1773> -> recurrir<8441> -> demanda<2935> -> petición<7596> -> cláusula<2053>
2	21	10	clavo<2063> -> clavero<2060> -> clavo<2063>
2	2	1	claxon<2064> -> bocina<1342> -> claxon<2064>
12	8	5	clérigo<2065> -> docto<3464> -> sabio<8893> -> prudente<8133> -> medida<6519> -> moderación<6619> -> extremo<4388> -> campo<1611> -> laborable<5908> -> festivo<4507> -> civil<2033> -> eclesiástico<3540> -> clérigo<2065>
8	8	7	cliente<2067> -> asiduidad<956> -> constancia<2437> -> firmeza<4564> -> estribillo<4218> -> muletilla<6761> -> bastón<1253> -> servir<9121> -> cliente<2067>
2	9	5	clorofila<2076> -> planta<7694> -> clorofila<2076>
2	13	4	club<2078> -> círculo<2013> -> club<2078>
2	9	5	coagular<2081> -> suero<9380> -> coagular<2081>
2	34	6	cobra<2089> -> naja<6816> -> cobra<2089>
4	14	9	cobre<2091> -> pardo<7301> -> mulato<6759> -> cobrizo<2092> -> cobre<2091>
2	22	1	coca<2094> -> cocaína<2095> -> coca<2094>
2	7	1	cocotero<2108> -> coco<2107> -> cocotero<2108>
2	9	3	codo<2114> -> antebrazo<642> -> codo<2114>
2	19	1	codorniz<2115> -> faisán<4408> -> codorniz<2115>
2	9	6	cofradía<2116> -> congregación<2383> -> cofradía<2116>
2	29	22	coger<2117> -> agarrar<299> -> coger<2117>
8	10	6	cohesión<2123> -> molécula<6645> -> puro<8201> -> construcción<2447> -> madera<6204> -> sustancia<9447> -> discurso<3378> -> coherencia<2121> -> cohesión<2123>
2	53	39	cola<2130> -> rabadilla<8246> -> cola<2130>
2	12	6	colaborar<2133> -> contribuir<2516> -> colaborar<2133>
6	15	20	colectivo<2143> -> referencia<8463> -> informe<5550> -> comunicación<2285> -> transporte<9837> -> empresa<3698> ->

<i>LC</i>	<i>FS</i>	<i>FE</i>	<i>Ciclo</i>
			colectivo<2143>
6	5	9	colegio<2144> -> centro<1876> -> educativo<3563> -> educar<3562> -> moral<6691> -> interno<5717> -> colegio<2144>
10	12	3	cólera<2146> -> calambre<1539> -> descarga<3052> -> descargar<3053> -> barreno<1234> -> barrena<1233> -> peñasco<7449> -> acceso<93> -> arrebató<870> -> furor<4754> -> cólera<2146>
4	8	6	colgado<2148> -> frustrar<4716> -> propósito<8082> -> móvil<6736> -> colgado<2148>
2	23	16	colgar<2151> -> ahorcar<354> -> colgar<2151>
2	14	5	colmillo<2156> -> elefante<3608> -> colmillo<2156>
3	6	7	colocación<2157> -> situación<9200> -> disposición<3413> -> colocación<2157>
2	13	68	colocar<2159> -> invertir<5771> -> colocar<2159>
12	33	7	colonia<2162> -> perfume<7511> -> despedir<3183> -> fórmula<4638> -> cilindrada<1989> -> cúbico<2752> -> cubo<2755> -> algebraico<434> -> álgebra<433> -> reducción<8457> -> indio<5469> -> poblador<7737> -> colonia<2162>
2	24	217	color<2164> -> colorido<2169> -> color<2164>
10	15	4	colorear<2168> -> madurar<6210> -> grano<4933> -> racimo<8251> -> prendido<7905> -> pelo<7437> -> opaco<7095> -> oscuro<7184> -> claro<2044> -> iluminar<5285> -> colorear<2168>
11	37	31	columna<2170> -> sostén<9325> -> sujeto<9392> -> mundo<6773> -> esfera<4039> -> radio<8264> -> espontáneo<4120> -> toro<9756> -> mamífero<6266> -> vertebrado<10173> -> vértebra<10172> -> columna<2170>
2	34	4	coma<2172> -> decimal<2853> -> coma<2172>
6	10	18	combate<2177> -> contradicción<2497> -> incompatibilidad<5411> -> calidad<1559> -> moral<6691> -> frente<4694> -> combate<2177>
2	5	9	combatir<2179> -> pelear<7432> -> combatir<2179>
6	2	1	combinatorio<2183> -> combinación<2180> -> sustancia<9447> -> conversación<2536> -> comunicación<2285> -> código<2112> -> combinatorio<2183>
2	15	17	combustible<2184> -> carbón<1696> -> combustible<2184>
2	15	5	comedia<2186> -> cómico<2208> -> comedia<2186>
2	29	34	comer<2196> -> comida<2209> -> comer<2196>
8	6	30	comercial<2197> -> aceptación<114> -> aceptar<115> -> pago<7229> -> campo<1611> -> distancia<3418> -> unir<9967> -> compañía<2220> -> comercial<2197>
2	15	19	comercio<2201> -> comerciante<2199> -> comercio<2201>
2	6	22	cometer<2204> -> comisión<2213> -> cometer<2204>
2	23	3	cómic<2207> -> globo<4875> -> cómic<2207>
2	8	1	comisaría<2211> -> comisario<2212> -> comisaría<2211>

<i>LC</i>	<i>FS</i>	<i>FE</i>	<i>Ciclo</i>
4	13	7	comodidad<2216> -> disposición<3413> -> salud<8942> -> bienestar<1306> -> comodidad<2216>
2	14	6	cómodo<2217> -> comfortable<2375> -> cómodo<2217>
7	24	5	compacto<2218> -> digital<3325> -> extracto<4374> -> expediente<4326> -> problema<7997> -> aclarar<135> -> denso<2954> -> compacto<2218>
2	5	15	comparar<2222> -> cotejar<2655> -> comparar<2222>
7	18	5	comparativo<2223> -> comparación<2221> -> retórica<8730> -> persuadir<7563> -> aceptar<115> -> pago<7229> -> cual<2736> -> comparativo<2223>
4	24	7	compás<2229> -> cadencia<1521> -> danza<2828> -> bailar<1164> -> compás<2229>
3	8	8	compensar<2237> -> neutralizar<6874> -> contrarrestar<2507> -> compensar<2237>
2	15	7	competencia<2238> -> rivalidad<8800> -> competencia<2238>
2	12	47	competición<2241> -> trofeo<9907> -> competición<2241>
9	7	6	competir<2242> -> logro<6163> -> ganancia<4794> -> obtener<7016> -> partir<7333> -> leña<6010> -> paliza<7249> -> derrota<2999> -> contrario<2506> -> competir<2242>
9	8	7	complacer<2246> -> acceder<92> -> favor<4459> -> concesión<2302> -> dominio<3488> -> libre<6051> -> riguroso<8785> -> duro<3532> -> cruel<2715> -> complacer<2246>
13	20	7	complejo<2248> -> subconsciente<9334> -> consciente<2410> -> pleno<7723> -> quiniela<8239> -> pronosticar<8058> -> futuro<4761> -> enunciación<3894> -> enunciar<3896> -> claro<2044> -> iluminar<5285> -> templo<9592> -> imaginario<5295> -> complejo<2248>
2	5	7	completar<2252> -> perfeccionar<7504> -> completar<2252>
3	9	38	completo<2253> -> rezo<8771> -> oración<7117> -> completo<2253>
2	30	63	composición<2261> -> redacción<8444> -> composición<2261>
2	10	7	compostura<2263> -> arreglo<873> -> compostura<2263>
2	4	9	comprar<2266> -> adquirir<243> -> comprar<2266>
2	13	76	comprender<2268> -> entender<3857> -> comprender<2268>
4	7	10	comprensión<2270> -> entender<3857> -> profundidad<8029> -> penetración<7457> -> comprensión<2270>
9	5	13	comprobar<2276> -> asegurar<937> -> infundir<5556> -> moral<6691> -> frente<4694> -> tropa<9914> -> componer<2258> -> corregir<2601> -> revisar<8757> -> comprobar<2276>
3	19	6	comprometer<2277> -> responsabilizar<8692> -> responsable<8693> -> comprometer<2277>
7	20	13	compromiso<2279> -> promesa<8048> -> triunfar<9904> -> vencer<10110> -> prevalecer<7955> -> arraigar<862> ->

<i>LC</i>	<i>FS</i>	<i>FE</i>	<i>Ciclo</i>
			fianza<4511> -> compromiso<2279>
2	41	116	compuesto<2281> -> margarita<6351> -> compuesto<2281>
2	28	302	común<2284> -> compartir<2228> -> común<2284>
8	21	28	comunicar<2287> -> consultar<2453> -> deliberar<2922> -> antes<654> -> anterior<652> -> consonante<2435> -> letra<6024> -> carta<1750> -> comunicar<2287>
6	18	49	comunidad<2288> -> calidad<1559> -> moral<6691> -> frente<4694> -> guerra<4989> -> nación<6808> -> comunidad<2288>
5	12	9	concebir<2292> -> formular<4639> -> fórmula<4638> -> receta<8370> -> modo<6629> -> concebir<2292>
2	6	3	concejal<2294> -> ayuntamiento<1143> -> concejal<2294>
4	24	36	concepto<2300> -> imposibilidad<5336> -> enfermedad<3780> -> orgánico<7138> -> concepto<2300>
2	16	7	concertar<2301> -> ajustar<371> -> concertar<2301>
2	24	17	concha<2303> -> ostra<7190> -> concha<2303>
2	18	13	conciencia<2304> -> moral<6691> -> conciencia<2304>
2	2	2	concisión<2309> -> brevedad<1415> -> concisión<2309>
2	4	4	concluir<2312> -> rematar<8560> -> concluir<2312>
4	8	8	conclusión<2313> -> fin<4549> -> consumación<2456> -> total<9773> -> conclusión<2313>
16	9	4	concurrir<2325> -> contribuir<2516> -> fin<4549> -> consumación<2456> -> cumplimiento<2795> -> obsequio<6998> -> agasajo<301> -> agasajar<300> -> halagar<5029> -> alegría<419> -> andaluz<589> -> caracterizado<1692> -> acuerdo<183> -> estamento<4151> -> condición<2336> -> encontrar<3748> -> concurrir<2325>
8	8	14	concurso<2326> -> puesto<8157> -> ambulante<534> -> fijo<4535> -> > invariable<5762> -> constante<2438> -> persistente<7551> -> mantener<6314> -> concurso<2326>
8	9	3	condensación<2331> -> molécula<6645> -> libre<6051> -> dispensar<3409> -> leve<6034> -> pesado<7580> -> cargazón<1717> -> nube<6948> -> condensación<2331>
7	14	3	condensar<2332> -> resumir<8714> -> sintetizar<9191> -> breve<1414> -> extensión<4365> -> extender<4363> -> espeso<4095> -> condensar<2332>
10	3	2	condescender<2334> -> acceder<92> -> favor<4459> -> empujar<3701> -> puesto<8157> -> camuflar<1612> -> engañar<3792> -> engatusar<3798> -> mimo<6579> -> condescendencia<2333> -> condescender<2334>
10	3	1	condescendiente<2335> -> acceder<92> -> favor<4459> -> empujar<3701> -> puesto<8157> -> camuflar<1612> -> engañar<3792> -> engatusar<3798> -> mimo<6579> ->

<i>LC</i>	<i>FS</i>	<i>FE</i>	<i>Ciclo</i>
			condescendencia<2333> -> condescendiente<2335>
2	10	25	conducir<2342> -> guiar<4995> -> conducir<2342>
7	10	40	conducta<2343> -> manejar<6288> -> mano<6307> ->
			jugador<5872> -> deportivo<2973> -> deportividad<2972> ->
			comportamiento<2259> -> conducta<2343>
2	8	5	confección<2350> -> hechura<5051> -> confección<2350>
7	13	8	confesar<2354> -> penitencia<7463> -> inquisición<5622> ->
			herejía<5076> -> sostenido<9327> -> entonar<3869> -> cantar<1651>
			-> confesar<2354>
2	18	3	confesión<2355> -> penitencia<7463> -> confesión<2355>
3	10	3	confesor<2356> -> penitente<7466> -> penitencia<7463> ->
			confesor<2356>
2	9	21	confianza<2357> -> seguridad<9036> -> confianza<2357>
2	24	1	confidente<2361> -> soplón<9300> -> confidente<2361>
2	17	4	confirmación<2365> -> confirmar<2366> -> confirmación<2365>
2	12	14	conforme<2373> -> acorde<155> -> conforme<2373>
2	6	20	conformidad<2374> -> resignación<8666> -> conformidad<2374>
2	5	17	confuso<2378> -> oscuro<7184> -> confuso<2378>
2	18	6	congreso<2385> -> diputado<3351> -> congreso<2385>
2	5	7	conjetura<2387> -> indicio<5462> -> conjetura<2387>
2	7	2	conjugación<2388> -> conjugación<2388> -> conjugar<2389>
2	13	10	conjunción<2390> -> invariable<5762> -> conjunción<2390>
30	29	2	conjuntivo<2392> -> locución<6154> -> unitario<9969> ->
			tender<9599> -> cable<1504> -> maroma<6364> -> circo<2008> ->
			fiera<4525> -> indómito<5486> -> domar<3481> -> bravura<1407> -
			> bravata<1404> -> amenaza<536> -> amenazar<537> ->
			anunciar<674> -> propaganda<8066> -> mercado<6339> ->
			marca<6338> -> certificar<1912> -> postal<7820> ->
			fotografía<4658> -> recogido<8401> -> retirado<8722> ->
			apartado<697> -> decreto<2870> -> cardenal<1705> ->
			plumaje<7730> -> pluma<7729> -> barbilla<1221> ->
			cartílago<1757> -> con
2	21	558	conjunto<2393> -> clase<2046> -> conjunto<2393>
6	9	2	conmoción<2397> -> perturbación<7571> -> estación<4141> ->
			fenómeno<4489> -> sensacional<9077> -> impresionante<5347> ->
			conmoción<2397>
2	4	13	conmover<2398> -> emocionar<3674> -> conmover<2398>
3	43	8	cono<2400> -> lava<5976> -> volcán<10294> -> cono<2400>
2	21	118	conocer<2402> -> entender<3857> -> conocer<2402>
2	18	72	conocimiento<2404> -> entendimiento<3859> ->
			conocimiento<2404>
3	11	10	conquistar<2406> -> conseguir<2415> -> pretender<7950> ->
			conquistar<2406>

<i>LC</i>	<i>FS</i>	<i>FE</i>	<i>Ciclo</i>
6	23	11	consagrar<2408> -> levantar<6032> -> sonar<9289> -> onda<7087> -> elevación<3615> -> hostia<5208> -> consagrar<2408>
3	3	4	consecución<2411> -> conseguir<2415> -> pretender<7950> -> consecución<2411>
7	8	6	consentimiento<2419> -> contratar<2512> -> contrata<2511> -> asegurar<937> -> infundir<5556> -> moral<6691> -> aprobación<761> -> consentimiento<2419>
9	12	5	consentir<2420> -> indulgente<5494> -> disculpa<3376> -> reproche<8629> -> censura<1864> -> artístico<912> -> arte<894> -> copiar<2557> -> permitir<7534> -> consentir<2420>
6	14	5	conservador<2422> -> vigente<10217> -> ordenanza<7132> -> instituto<5660> -> centro<1876> -> izquierda<5820> -> conservador<2422>
2	11	25	conservar<2423> -> mantener<6314> -> conservar<2423>
7	9	20	consideración<2425> -> deferencia<2885> -> moderación<6619> -> extremo<4388> -> campo<1611> -> tierra<9675> -> considerado<2426> -> consideración<2425>
2	7	56	considerar<2427> -> reflexionar<8473> -> considerar<2427>
19	4	3	consolar<2433> -> tristeza<9902> -> triste<9901> -> pesadumbre<7581> -> desazón<3042> -> inquietud<5619> -> ambición<529> -> éxito<4317> -> aceptación<114> -> aceptar<115> -> letra<6024> -> cantar<1651> -> oler<7069> -> pasa<7337> -> secar<9002> -> vitalidad<10270> -> expansión<4321> -> solaz<9249> -> consuelo<2450> -> consolar<2433>
7	11	51	constar<2439> -> registrado<8507> -> marca<6338> -> homologar<5165> -> confirmar<2366> -> aprobado<762> -> baja<1167> -> constar<2439>
4	22	13	constitución<2443> -> elemento<3614> -> cuatro<2749> -> figura<4529> -> constitución<2443>
9	2	113	constituir<2444> -> componer<2258> -> plana<7683> -> superficie<9409> -> extensión<4365> -> extender<4363> -> espeso<4095> -> macizo<6201> -> planta<7694> -> constituir<2444>
7	3	3	constitutivo<2445> -> definir<2891> -> claridad<2042> -> claro<2044> -> cuadro<2731> -> decorado<2864> -> elemento<3614> -> constitutivo<2445>
2	19	4	consulta<2452> -> consultorio<2455> -> consulta<2452>
9	5	5	consumido<2457> -> flaco<4571> -> carne<1729> -> personal<7555> -> elemento<3614> -> cuatro<2749> -> figura<4529> -> carta<1750> -> restaurante<8701> -> consumido<2457>
8	11	15	consumir<2458> -> cambiar<1594> -> rumbo<8877> -> trazado<9868> -> recorrido<8422> -> recorrer<8421> -> leer<5991> -> emplear<3695> -> consumir<2458>
2	22	29	contacto<2462> -> comunicación<2285> -> contacto<2462>

<i>LC</i>	<i>FS</i>	<i>FE</i>	<i>Ciclo</i>
14	6	4	contemplación<2469> -> deferencia<2885> -> adhesión<212> -> molécula<6645> -> puro<8201> -> fumar<4733> -> despedir<3183> -> lanzar<5950> -> flor<4587> -> mejor<6453> -> artículo<904> -> luna<6187> -> lente<6014> -> visión<10257> -> contemplación<2469>
10	5	1	contemplativo<2471> -> acostumbrar<159> -> soler<9259> -> domingo<3487> -> seguir<9034> -> cursar<2810> -> centro<1876> -> comedor<2190> -> dicha<3291> -> felicidad<4478> -> contemplativo<2471>
2	5	94	contener<2474> -> reprimir<8625> -> contener<2474>
2	3	17	continuar<2489> -> durar<3530> -> continuar<2489>
15	9	23	continuo<2491> -> remisión<8568> -> remitir<8571> -> liberar<6045> -> libre<6051> -> preso<7935> -> espera<4089> -> aguardar<336> -> futuro<4761> -> enunciación<3894> -> enunciar<3896> -> claro<2044> -> iluminar<5285> -> bañar<1194> -> río<8791> -> continuo<2491>
2	22	8	contracción<2495> -> músculo<6789> -> contracción<2495>
2	9	11	contradecir<2496> -> contradicción<2497> -> contradecir<2496>
8	10	16	contraer<2498> -> nervio<6867> -> duro<3532> -> flexibilidad<4583> -> ceder<1831> -> cesar<1916> -> emperador<3690> -> vasallo<10066> -> contraer<2498>
3	14	2	contrafuerte<2499> -> empuje<3702> -> estribo<4219> -> contrafuerte<2499>
10	9	10	contrariedad<2505> -> logro<6163> -> ganancia<4794> -> obtener<7016> -> partir<7333> -> leña<6010> -> paliza<7249> -> derrota<2999> -> vencer<10110> -> dificultad<3315> -> contrariedad<2505>
2	21	3	contraste<2510> -> contrastar<2509> -> contraste<2510>
4	8	27	contrato<2514> -> acuerdo<183> -> estamento<4151> -> profesional<8025> -> contrato<2514>
2	11	6	contribución<2515> -> contribuir<2516> -> contribución<2515>
12	7	10	convencer<2525> -> gustar<5005> -> paladar<7239> -> sensibilidad<9080> -> calidad<1559> -> moral<6691> -> interno<5717> -> rama<8273> -> planta<7694> -> célula<1849> -> envolvente<3913> -> convincente<2542> -> convencer<2525>
2	5	2	convencimiento<2526> -> persuasión<7564> -> convencimiento<2526>
2	5	28	conveniente<2530> -> adecuado<197> -> conveniente<2530>
2	4	12	convenio<2531> -> pacto<7216> -> convenio<2531>
2	4	3	convidar<2541> -> invitar<5782> -> convidar<2541>
2	7	2	convite<2543> -> invitar<5782> -> convite<2543>
2	4	8	convocar<2545> -> citar<2027> -> convocar<2545>
11	21	2	coordinación<2550> -> ion<5788> -> negativa<6854> ->

LC FS FE Ciclo

			negación<6852> -> pronombre<8057> -> contexto<2481> -> secuencia<9013> -> seguir<9034> -> cursar<2810> -> centro<1876> -> coordinado<2551> -> coordinación<2550>
12	30	12	copa<2553> -> constelación<2441> -> estrella<4212> -> lunar<6188> -> variable<10054> -> pronombre<8057> -> puro<8201> -> mero<6511> -> rechoncho<8375> -> bajo<1170> -> toca<9713> -> ala<374> -> copa<2553>
10	15	2	copista<2558> -> literatura<6115> -> repertorio<8604> -> sumario<9398> -> preparación<7910> -> sustancia<9447> -> sujeto<9392> -> proceso<8006> -> dictar<3299> -> escriba<4012> -> copista<2558>
2	14	1	copo<2559> -> nieve<6884> -> copo<2559>
2	12	3	coraza<2564> -> blindaje<1333> -> coraza<2564>
9	12	1	corchete<2567> -> broche<1431> -> aguja<344> -> torre<9759> -> destilación<3224> -> soltar<9272> -> confinamiento<2364> -> recluir<8392> -> encerrar<3736> -> corchete<2567>
2	24	3	corcho<2568> -> alcornoque<409> -> corcho<2568>
11	8	3	córnea<2578> -> duro<3532> -> cinco<1996> -> figura<4529> -> superficie<9409> -> extensión<4365> -> extender<4363> -> espeso<4095> -> pesado<7580> -> cargazón<1717> -> nube<6948> -> córnea<2578>
2	5	4	corola<2582> -> flor<4587> -> corola<2582>
2	34	8	corona<2583> -> aureola<1077> -> corona<2583>
2	5	2	coronel<2584> -> regimiento<8502> -> coronel<2584>
2	9	1	coronilla<2585> -> tonsura<9733> -> coronilla<2585>
6	8	22	corporación<2586> -> defensa<2882> -> acusado<189> -> proceso<8006> -> sentencia<9088> -> consejo<2417> -> corporación<2586>
2	3	1	corpuscular<2591> -> corpúsculo<2592> -> corpuscular<2591>
2	13	15	correa<2594> -> cinturón<2006> -> correa<2594>
7	21	1	corredera<2598> -> postigo<7825> -> gozne<4900> -> espiga<4100> -> espina<4101> -> circo<2008> -> carro<1745> -> corredera<2598>
2	24	8	corredor<2600> -> avestruz<1128> -> corredor<2600>
6	22	7	correlación<2602> -> pertinente<7569> -> lija<6071> -> escama<3972> -> sobrepuesto<9218> -> cual<2736> -> correlación<2602>
6	11	17	correspondencia<2607> -> elemento<3614> -> cuatro<2749> -> figura<4529> -> musical<6794> -> armonioso<852> -> correspondencia<2607>
2	17	3	corro<2614> -> círculo<2013> -> corro<2614>
2	5	8	corromper<2617> -> pervertir<7575> -> corromper<2617>
2	13	1	corso<2622> -> corsario<2621> -> corso<2622>
10	8	13	cortado<2624> -> apocado<732> -> tímido<9681> -> timidez<9680>

LC FS FE Ciclo

-> seguridad<9036> -> firmeza<4564> -> estribillo<4218> ->
muletilla<6761> -> muleta<6760> -> mano<6307> -> cortado<2624>
4 5 16 cortante<2625> -> cuchillo<2760> -> corte<2627> -> filo<4543> ->
cortante<2625>
2 31 50 cortar<2626> -> recortar<8424> -> cortar<2626>
2 3 14 cortés<2630> -> educado<3561> -> cortés<2630>
4 13 26 cortesía<2631> -> prórroga<8088> -> obligatorio<6991> ->
cumplimiento<2795> -> cortesía<2631>
8 18 34 corteza<2632> -> cubierta<2753> -> editorial<3559> ->
artículo<904> -> determinante<3261> -> partir<7333> -> leña<6010>
-> madera<6204> -> corteza<2632>
2 12 32 corto<2634> -> escaso<3988> -> corto<2634>
3 16 1695 cosa<2637> -> inanimado<5374> -> vida<10206> -> cosa<2637>
2 13 14 coser<2639> -> aguja<344> -> coser<2639>
2 9 1 cosquilleo<2644> -> hormiguelo<5190> -> cosquilleo<2644>