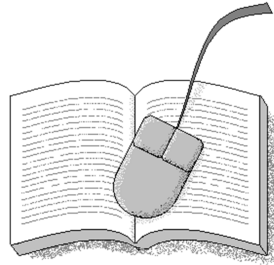


**Instituto Politécnico Nacional**  
**Centro de Investigación en Computación**



Laboratorio de Lenguaje Natural y Procesamiento de Texto  
*Detección automática de primitivos semánticos con algoritmos  
bioinspirados*

**T E S I S**

QUE PARA OBTENER EL GRADO DE  
DOCTOR EN CIENCIAS DE LA COMPUTACIÓN

PRESENTA

**M. en C. OBDULIA PICHARDO LAGUNAS**

DIRECTORES DE TESIS

**DR. GRIGORI SIDOROV**

**DRA. NARELI CRUZ CORTÉS**

México, D. F. enero 2013



**INSTITUTO POLITÉCNICO NACIONAL  
SECRETARÍA DE INVESTIGACIÓN Y POSGRADO**

SIP-14 BIS

**ACTA DE REVISIÓN DE TESIS**

En la Ciudad de     México, D.F.     siendo las     16:00     horas del día     14     del mes de     Diciembre     de     2012     se reunieron los miembros de la Comisión Revisora de la Tesis, designada por el Colegio de Profesores de Estudios de Posgrado e Investigación del:

**Centro de Investigación en Computación**

para examinar la tesis titulada:

**“DETECCIÓN AUTOMÁTICA DE PRIMITIVOS SEMÁNTICOS CON ALGORITMOS BIOINSPIRADOS”**

Presentada por la alumna:

**PICHARDO**  
Apellido paterno

**LAGUNAS**  
Apellido materno

**OBDULIA**  
Nombre(s)

Con registro: 

A	0	9	0	1	8	7
---	---	---	---	---	---	---

aspirante de: **DOCTORADO EN CIENCIAS DE LA COMPUTACIÓN**

Después de intercambiar opiniones los miembros de la Comisión manifestaron **APROBAR LA TESIS**, en virtud de que satisface los requisitos señalados por las disposiciones reglamentarias vigentes.

**LA COMISIÓN REVISORA**

Directores de tesis

Dr. Grigori Sidorov

Dra. Nareli Cruz Cortés

Dr. Sergio Suárez Guerra

Dr. Alexander Gelbukh

Dra. Sofía Natalia Galicia Haro

PRESIDENTE DEL COLEGIO DE PROFESORES

Dr. Luis Alfonso Villa Vargas



INSTITUTO POLITÉCNICO NACIONAL  
CENTRO DE INVESTIGACION  
EN COMPUTACION  
DIRECCION




**INSTITUTO POLITÉCNICO NACIONAL**  
**SECRETARÍA DE INVESTIGACIÓN Y POSGRADO**

**CARTA CESIÓN DE DERECHOS**

En la Ciudad de México D.F. el día 10 del mes enero del año 2013, el (la) que suscribe Obdulia Pichardo Lagunas alumno (a) del Programa de Doctorado en Ciencias de la Computación con número de registro A090187 adscrito a Centro de Investigación en Computación, manifiesta que es autor (a) intelectual del presente trabajo de Tesis bajo la dirección de Dr. Grigori Sidorov y Dra. Nareli Cruz Cortés y cede los derechos del trabajo intitulado Detección Automática de Primitivas Semánticas con algoritmos bioinspirados, al Instituto Politécnico Nacional para su difusión, con fines académicos y de investigación.

Los usuarios de la información no deben reproducir el contenido textual, gráficas o datos del trabajo sin el permiso expreso del autor y/o director del trabajo. Este puede ser obtenido escribiendo a la siguiente dirección opichardola@ipn.mx. Si el permiso se otorga, el usuario deberá dar el agradecimiento correspondiente y citar la fuente del mismo.

  
Obdulia Pichardo Lagunas

Nombre y firma

# Resumen

Cualquier diccionario explicativo tradicional inevitablemente contiene ciclos en sus definiciones, es decir, si una palabra es definida en el diccionario y después se usa en una definición, siempre existe un camino en este diccionario que regresa a la misma palabra. Un ejemplo de un ciclo de longitud dos: “pacto es convenio”, “convenio es tratado”, “tratado es pacto”: en dos pasos regresamos a la misma palabra. En un buen diccionario los ciclos son largos, pero son inevitables. Un diccionario semántico computacional (destinado para el uso de las computadoras) no puede contener ciclos en sus definiciones sin que éstos afecten la capacidad de inferencia lógica de los sistemas computacionales. Denominamos primitivas semánticas a un conjunto de palabras que de ser eliminadas del diccionario lo mantendría sin ciclos, es decir, esas palabras no tendrán definición en el diccionario, y en este sentido son primitivas. En esta tesis, nuestra meta es mantener la mayor cantidad de palabras en el diccionario obteniendo un número mínimo de las primitivas semánticas. Presentamos un método que extrae el conjunto de primitivas más pequeño hasta ahora. Para eso utilizamos la representación del diccionario como un grafo dirigido y aplicamos algoritmos bioinspirados que determinan el orden en que el grafo debe ser construido.

# Abstract

Any traditional explanatory dictionary inevitably has cycles in its definitions, i.e., if a word is defined in the dictionary and then is used in a definition, there always exists a path in this dictionary that returns to the same word. For example, a cycle of length two: "pacto es convenio (pact is an agreement)", "convenio es tratado (agreement is a treaty)," "tratado es pacto (treaty is a pact)": there are two steps to come back to the same word. In a good dictionary these cycles are long, but they are inevitable. A computational semantic dictionary that is meant for the use by computers cannot contain cycles in their definitions because they affect the ability of logical inference of computer systems. We call semantic primitives a set of words that should be eliminated from the dictionary so that the dictionary would contain no cycles. In this way, these words (semantic primitives) have no definition in the dictionary, and in this sense they are primitive. In this thesis, our goal is to keep as many words in the dictionary as possible, i.e., to obtain the smallest number of semantic primitives. We present a method that constructs the smallest set of primitives as compared to other methods. We use the dictionary representation as a directed graph and apply a differential evolution algorithm that determines the order in which the graph is constructed. We evaluate our results using dictionary frequencies and Internet frequencies of words.

# Agradecimientos

A mi madre Irene Sánchez Torres.

A mis hermanas Paty, Adriana, Perla y Zamná.

La realización de este trabajo fue posible gracias al apoyo de personas e instituciones a las que menciono a continuación:

Mi muy especial agradecimiento a mis asesores el Dr. Grigori Sidorov y la Dra. Nareli Cruz Cortés por compartir conmigo su tiempo y conocimientos, por brindarme siempre su apoyo y hacerme objeto de su paciencia.

Agradezco también a los Dres. Alexander Gelbukh, Sergio Suarez Guerra y Sofía Galicia Haro miembros del comité tutorial por sus valiosas observaciones.

Agradezco al M. Sabino Miranda Jiménez y al Dr. Noé Castro Sánchez por sus valiosas aportaciones.

Al Instituto Politécnico Nacional por darme la oportunidad de continuar con mi preparación.

Hago extensivo mi agradecimiento al Consejo Nacional de Ciencia y Tecnología (CONACYT) que me proporcionó los medios económicos necesarios para poder cursar los estudios de doctorado.

## TABLA DE CONTENIDO

Resumen.....	4
Abstract.....	5
Agradecimientos .....	6
Capítulo 1 Introducción .....	11
1.1 Motivación .....	11
1.2 Descripción del problema.....	13
1.3 Hipótesis.....	13
1.4 Objetivos .....	14
1.5 Organización del documento.....	14
Capítulo 2 Marco teórico .....	16
2.1 Lingüística.....	16
2.2 La lingüística computacional .....	18
2.3 Lexicografía computacional .....	19
2.4 Primitivas semánticas.....	21
2.5 Teoría de grafos.....	24
2.6 Computación evolutiva .....	27
Capítulo 3 Estado del arte .....	44
3.1 LDOCE como Red semántica .....	44
3.2 Extracción automática de primitivas semánticas con métodos aleatorios.....	47
Capítulo 4 Método propuesto.....	51
4.1 Selección de diccionarios .....	52
4.2 Preprocesamiento de los diccionarios .....	54
4.3 Aplicación de los algoritmos evolutivos .....	58

4.1	Configuración del algoritmo genético.....	60
4.2	Configuración del algoritmo de evolución diferencial.....	62
4.3	Descripción de los experimentos.....	63
Capítulo 5	Discusión de los resultados .....	66
Capítulo 6	Conclusiones y trabajo futuro .....	76
6.1	Conclusiones .....	76
6.2	Aportaciones.....	77
6.3	Trabajo futuro.....	77
Referencias.....		78



## ÍNDICE DE TABLAS

Tabla 1. Primitivas semánticas [6].....	23
Tabla 2. Terminología utilizada en computación evolutiva.....	29
Tabla 3. Variaciones de ED .....	39
Tabla 4. Configuración de parámetros del Algoritmo Genético.....	62
Tabla 5. Configuraciones de parámetros del Algoritmo de Evolución Diferencial.....	63
Tabla 6. Configuraciones con mejores resultados. ....	66
Tabla 7. Resultados diccionario Anaya con Algoritmo Genético.....	68
Tabla 8. Resultados diccionario Anaya con AED.....	69
Tabla 9. Diccionario Anaya Algoritmo de Evolución Diferencial .....	70
Tabla 10. Resultados obtenidos para Anaya: comparación con el conjunto Rivera-Loza .....	70
Tabla 11. Comparación entre AG y AED con RAE .....	70
Tabla 12. Resultados obtenidos para RAE: comparación con el conjunto Rivera-Loza .....	71
Tabla 13. Primitivas semánticas con mayor frecuencia.....	74

## ÍNDICE DE ILUSTRACIONES

Ilustración 1. Lingüística computacional .....	18
Ilustración 2. LC y sus relaciones como ciencia cognitiva .....	19
Ilustración 3. Grafos simples .....	25
Ilustración 4. Matriz de adyacencia .....	26
Ilustración 5. Lista de adyacencia .....	26
Ilustración 6. Trayectoria dirigida.....	27
Ilustración 7. Algoritmo evolutivo básico.....	28
Ilustración 8. Representación de la población .....	31
Ilustración 9. Algoritmo de evolución diferencial .....	39
Ilustración 10. Nodos de la red semántica. ....	493
Ilustración 11. Representación del diccionario .....	507
Ilustración 12. Grafo tipo1 .....	508
Ilustración 13. Grafo tipo 2.....	50
Ilustración 14. Modelo computacional.....	52
Ilustración 15. Preprocesamiento del diccionario .....	54
Ilustración 16. Información adicional en la definición .....	564
Ilustración 17. Ejemplo de prefijo .....	56
Ilustración 18. Palabra que genera lazo .....	56
Ilustración 19. Palabras sin uso en las definiciones .....	56
Ilustración 20. Comparación de los resultados del AG y AED para diccionario Anaya .....	67
Ilustración 21. Frecuencia en el diccionario Anaya.....	721
Ilustración 22. Frecuencia en Internet.....	72

# Capítulo 1

## Introducción

### 1.1 Motivación

El objetivo principal del procesamiento de lenguaje natural es la automatización de procesos lingüísticos tales como, traducción automática, generación automática de resúmenes, contestadores de preguntas, etc.

El lexicón es el conjunto de palabras propias de una lengua ordenadas según determinado criterio y es también el componente principal de cualquier aplicación de procesamiento de lenguaje natural (PLN). La mayoría de los lexicones utilizados en aplicaciones de PLN contienen un número limitado de palabras. Dado que no conocemos el tamaño ideal ni cuál es la mejor forma de representarlo, la construcción manual de un lexicón puede demandar muchos recursos, por ejemplo, en 1857 la construcción *Oxford English Dictionary (OED)* se esperaba que tomara 10 años, 27 años después en 1884 se publicó sólo el primer fascículo y el trabajo aún continúa. Situaciones como ésta ha llevado a muchos investigadores a considerar las versiones ya existentes de los diccionarios sobre todo las electrónicas, como fuentes potenciales de información léxica.

Aun así, sin importar cuál sea la fuente del lexicón sabemos que éste siempre tendrá los problemas característicos de cualquier diccionario, como la subjetividad en las definiciones derivada de la personalidad del lexicógrafo o los inevitables ciclos en las definiciones. Evitar la existencia de estos ciclos en los diccionarios utilizados para PLN es el objetivo de este trabajo.

Sabemos que cualquier diccionario explicativo tradicional inevitablemente contiene ciclos en sus definiciones. En un buen diccionario los ciclos son largos, pero son inevitables

Ahora bien, para poder utilizar un diccionario como herramienta computacional debemos excluir de éste los posibles ciclos en las definiciones ya que pueden afectar la capacidad de inferencia lógica de los sistemas computacionales. La teoría sobre la existencia de un conjunto de palabras—primitivas semánticas—a partir de las cuales se pueda definir el resto de las voces de una lengua plantea una posible solución al problema de los ciclos.

Los ciclos en un diccionario explicativo representan un gran problema para cualquier trabajo semántico que involucre algún tipo de razonamiento. Un ejemplo es la resolución de la ambigüedad en el sentido de las palabras. Este caso se presenta cuando una palabra tiene varios sentidos. Un método para encontrar el sentido preciso de una 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 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 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.

Sabemos que 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).

Marcar algunas palabras como "primitivas" y en cierto sentido eliminarlas del diccionario permitirá que la definición de cualquier otra palabra que se expresa con "primitivas" ya sea en uno o varios pasos forme parte de algún ciclo.

Posteriormente, se puede dar algún tratamiento especial a esas palabras primitivas, por ejemplo, sustituyéndolas por otro tipo de elementos comprensibles para los sistemas computacionales.

## **1.2 Descripción del problema**

Las definiciones en los diccionarios orientados a humanos inevitablemente tienen ciclos. Ejemplo: “*dar*”: *entregar, otorgar*. “*entregar*”: *dar o poner en poder de otro*.

En un diccionario orientado a lectores humanos los ciclos muy cortos no son deseables porque no proporcionan información suficiente al lector, pero ya que los ciclos son inevitables se espera que al menos éstos sean largos.

Consideramos primitivas semánticas las palabras que en uno o más pasos cierran un ciclo en las definiciones.

Para identificar estas palabras, se representó al diccionario como un grafo dirigido, marcando la relación existente entre cada entrada del diccionario y las palabras contenidas en su definición. Dada la dificultad de la búsqueda de ciclos en un grafo ya construido, el diccionario fue creado palabra por palabra verificando la existencia de ciclos en cada paso.

## **1.3 Hipótesis**

Dado que con diferentes órdenes de entrada al grafo se generan conjuntos de primitivas semánticas diferentes en tamaño y en características, se considera que el orden en que las palabras se ingresan al grafo dirigido es relevante.

Para obtener el menor y mejor conjunto de primitivas semánticas, se busca determinar la mejor permutación de entrada de las palabras al grafo.

## **1.4 Objetivos**

### ***Objetivo general***

Obtener un conjunto de primitivas semánticas de un diccionario explicativo en español utilizando los algoritmos bioinspirados, que por su construcción sea mínimo y que sea considerado un conjunto de calidad con respecto a su frecuencia.

### ***Objetivos específicos***

- Generar la matriz de adyacencia de dos diferentes diccionarios explicativos para español.
- Aplicar un algoritmo de evolución diferencial para la extracción de primitivas semánticas utilizando como representación del diccionario un grafo dirigido.
- Aplicar un algoritmo genético para la extracción de primitivas semánticas utilizando como representación del diccionario un grafo dirigido.
- Verificar la calidad de los conjuntos de primitivas obtenidos analizando su frecuencia en el mismo diccionario, en internet y en una lista traducida del vocabulario inglés Longman.

## **1.5 Organización del documento**

El resto de este trabajo está organizado como sigue.

En el capítulo 2 constituye el marco teórico necesario para el estudio del tema abordado en esta tesis, una introducción a la lingüística computacional, la teoría de grafos y la computación evolutiva.

En el capítulo 3 Proporciona una introducción a cerca del estado del arte desde el punto de vista de la Lingüística computacional. Menciona los trabajos realizados a la fecha entorno a la detección automática de primitivas semánticas y el trabajo con diccionarios.

En el capítulo 4 se presentan las principales características del modelo propuesto para la detección automática de primitivas semánticas.

En el capítulo 5 se presentan las conclusiones de esta tesis, las aportaciones y el trabajo futuro.

# Capítulo 2

## Marco teórico

### 2.1 Lingüística

Lingüística es la ciencia que se encarga del estudio de todas las manifestaciones del lenguaje humano. Esta ciencia está compuesta por diferentes ramas que permiten su interacción con otros campos de investigación como la lingüística computacional.

El análisis del lenguaje se divide en cinco niveles:

#### **Nivel Fonético/Fonológico**

Se encarga del estudio del fonema que es la unidad básica e indivisible del lenguaje. Proporciona información sobre el sistema de sonidos y la estructura de las palabras y las expresiones, los patrones de acentuación, la entonación, etc.

Los problemas para este nivel de análisis pueden ser los sistemas de reconocimiento de voz y síntesis de habla. Hay avances en ambos campos, ya existen sistemas capaces de reconocer las palabras pronunciadas en el micrófono —aunque en muchos casos con errores— y sistemas que si bien no suenan como humano hablan bastante bien.

#### **Nivel Morfológico**

Estudia las clases de morfemas y palabras, y su estructura. Los morfemas como unidades mínimas de la lengua que poseen significado. Este significado puede ser léxico o gramatical.

Los problemas de morfología computacional están relacionados con el desarrollo de sistemas de análisis y síntesis morfológica automática. El desarrollo de tales módulos es aún bastante engorroso porque hay que hacer grandes diccionarios de raíces (alrededor de cien mil). En general existe la metodología de tal desarrollo y existen sistemas funcionando para muchos idiomas [34].



## **Nivel Sintáctico**

Estudia la organización de las palabras en frases y oraciones. Clasifica las palabras en categorías gramaticales: artículos, sustantivos, adjetivos, verbos, pronombres, adverbios, preposiciones y conjunciones.

La sintaxis computacional debe tener métodos para análisis y síntesis automática, es decir, construir la estructura de frase, o generar la frase basándose en su estructura. El desarrollo de los analizadores sintácticos (también llamados *parsers*) todavía es un problema abierto, especialmente para los idiomas que no tienen un orden de palabras fijo, como el español.

## **Nivel Semántico**

El propósito de la semántica es “entender” la frase, para esto es necesario saber el sentido de todas las palabras e interpretar las relaciones sintácticas. Un resultado del análisis semántico deben ser redes semánticas, donde se representan todos los conceptos y las relaciones entre ellos y los grafos conceptuales. El problema es que se necesita saber cómo hacer la transformación de un árbol sintáctico en una red semántica. Esto aún no tiene una solución general.

La semántica tiene dos subdisciplinas llamadas lexicología y lexicografía. Éstas se encargan de definir los sentidos de las palabras y la creación de diccionarios. El problema principal es que siempre existe un círculo vicioso en las definiciones de las palabras, porque las palabras se definen a través de otras palabras. La semántica computacional puede ayudar buscando un conjunto de las palabras a través de las cuales se definirán las demás palabras: el vocabulario definidor. Otro problema específico es evaluar automáticamente la calidad de los diccionarios. Todos usamos los diccionarios y sabemos que hay tanto diccionarios buenos, como malos. Una aplicación importante del análisis semántico es la desambiguación automática de sentidos de palabras. Definir que sentidos se usan en un contexto dado.

## Nivel Pragmático

Estudia el análisis de las presuposiciones del hablante, o de las intenciones comunicativas que subyacen en una frase en particular contemplando el contexto extralingüístico.

## 2.2 La lingüística computacional

El lenguaje humano es con frecuencia alusivo y ambiguo. Alusivo, porque las palabras pueden incorporar referencias a múltiples niveles. Las personas, al leer o escuchar, nos hemos acostumbrado a detectar indicios y pistas. Pero los ordenadores no tienen ese sexto sentido. La ambigüedad es, quizás, un problema aun mayor. Muchas frases, hasta el 40% en ciertos tipos de textos, pueden resultar ambiguas para un ordenador, incluso aunque tengan sentido para un traductor humano, porque este tiene su conocimiento “extratextual” [35].

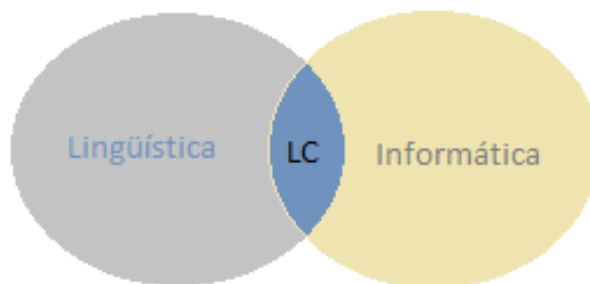
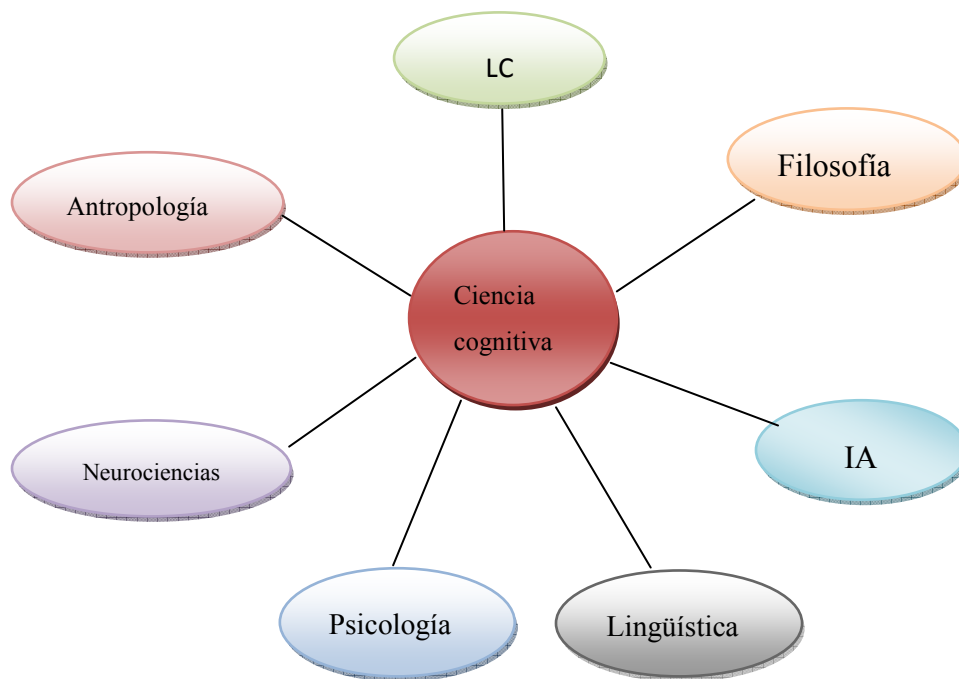


Ilustración 1. Lingüística computacional

La lingüística computacional (LC) es un campo interdisciplinario entre la lingüística y la informática, ver Ilustración 1. Cuyo objetivo es la elaboración de modelos computacionales que reproduzcan distintos aspectos del lenguaje humano. Utiliza las descripciones generadas para cada nivel del lenguaje y trata de representarlas como modelos computacionales que servirán como herramienta para la solución de tareas como el etiquetamiento morfológico (o *tagging*), el análisis sintáctico (o *parsing*), los procesos de interpretación semántica, la traducción automática, las técnicas de reconocimiento de voz o conversión de texto a voz, la recuperación inteligente de información, los sistemas de diálogo y sistemas expertos. Los modelos generados por la lingüística computacional para

describir el funcionamiento del lenguaje no tienen por qué reproducir exactamente el fenómeno del lenguaje. Únicamente constituyen una herramienta para acercarnos a la comprensión de éste.

Ya que el lenguaje es considerado parte fundamental del sistema cognitivo humano, se puede situar la LC en un marco de confluencia más amplio que el de la mera intersección de Lingüística e Informática: el que le proporciona en la actualidad la Ciencia que, tomando como punto de referencia el objetivo común de estudiar la mente humana, como se ve en la Ilustración 2, aglutina áreas del saber tan diversas como la Lingüística, la Psicología, la Neurociencia, la Antropología, la Filosofía y la propia Inteligencia Artificial, entre otras [36].



**Ilustración 2. LC y sus relaciones como ciencia cognitiva**

### **2.3 Lexicografía computacional**

La lexicografía computacional centra sus esfuerzos en la construcción de lexicones computacionales para el procesamiento de lenguaje natural. Los lexicones son los recursos

más importantes en la construcción de sistemas computacionales que posibilitan la interacción entre la máquina y el hombre.

Los diccionarios explicativos conforman la base de la descripción lexicográfica de un lenguaje, rigen el uso correcto y comprensión de una lengua, Son elaborados por equipos de profesionales que tratan de garantizar su calidad tarea que es muy difícil utilizando los métodos tradicionales. Esto se debe a que un diccionario es un sistema de elementos interrelacionados entre sí y al dinamismo característico del lenguaje. Como en cualquier sistema, su calidad sólo puede evaluarse considerando las interrelaciones que existen entre los diferentes elementos en este caso los vocablos.

En el ámbito computacional, los lexicones son parte importante en la construcción de sistemas informáticos. No se pueden construir sistemas de procesamiento de lenguaje natural sin antes diseñar lexicones que contengan información léxica detallada que servirán como apoyo en el procesamiento de la información. Tareas como la generación de textos, la traducción automática, el desarrollo de sistemas de reconocimiento de habla, los interfaces en lenguaje natural, etc., Han provocado una demanda constante de información léxica sin embargo, la construcción de un lexicón para su utilización en este tipo de tareas requiere una elaboración minuciosa.

Algunos de los diccionarios usados son los siguientes: *Oxford Advanced Learner's Dictionary of Current English (OALD)*, *The Collins Cobuild English Language Dictionary (COBUILD)*, *Longman Dictionary of Contemporary English (LDOCE)*, *Webster's Seventh Collegiate Dictionary (W7)*, *Merriam-Webster Pocket Dictionary (MWPD)*.

Los primeros trabajos realizados con los diccionarios electrónicos se dedicaron a estudiar frecuencias de palabras en las definiciones. Al mismo tiempo se llevaron a cabo investigaciones, relativas a las redes semánticas, se estaban empezando a estudiar los “enlaces” (*links*), “cadenas” (*chains*) y “círculos” (*circles*) que se forman en un diccionario a través de las palabras que se usan en sus definiciones, con vistas a la construcción automática de taxonomías. En esta línea de investigación, el trabajo de Karen Sparck-Jones, demostró que la “circularidad” debe, en principio, existir en un diccionario, ya que cada palabra usada en las definiciones ha de ser, a su vez, definida en el diccionario. Algunas de estas circularidades mantienen una distancia semántica reducida, como por

ejemplo las definiciones mutuas de “*good*” y “*excellent*”, y son por tanto fáciles de observar y asimilar por un lector humano, pero son muy difíciles de localizar a nivel formal y esto puede dificultar enormemente la labor de extracción de información de las definiciones, sobre todo si se aplican nociones empíricas de derivación circular [4].

La versión magnética del LDOCE contiene 41.000 entradas, con información adicional a la que se encuentra en la edición en papel. Sus autores afirman que las entradas han sido definidas usando un vocabulario “controlado” de 2.000 palabras y que las entradas tienen una sintaxis simple, lo que parece reducir la circularidad en las definiciones.

## **2.4 Primitivas semánticas**

Las palabras establecidas para la creación de definiciones son conocidas como vocabulario definidor. Algunos diccionarios como LDOCE y OALD crean la totalidad de sus definiciones con un vocabulario de un poco más de 2000 palabras. En un sentido más estricto podríamos denominar al vocabulario definidor como primitivas semánticas. Hasta ahora este conjunto de palabras es elegido de forma manual sin criterios bien definidos.

Un diccionario para el usuario humano tiene como propósito explicar la palabra, una forma de hacerlo puede ser utilizar una amplia variedad de palabras que puedan ser conocidas por el lector y que le faciliten la comprensión de la definición. Otro modo de aumentar la probabilidad de comprensión es usar, en las definiciones, sólo un número restringido de las palabras más simples y conocidas (vocabulario definidor). Evitar la existencia de círculos viciosos cortos en el sistema de definiciones ayudará al usuario a comprender la definición.

El metalenguaje semántico natural (NSM -*Natural Semantic Metalanguage*-) es una teoría desarrollada por Anna Wierzbicka y por Cliff Goddard [5], 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 consiste en describir un significado complejo en términos de otros más simples, esto es, para consignar el significado de una palabra compleja, se dará una paráfrasis formada por otras palabras más "simples" y más fáciles de entender que la original [6].

Según esta teoría, cada lenguaje tiene un conjunto de primitivas que es irreducible 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 sub-lenguaje con la misma expresividad que el lenguaje natural, la lista propuesta por Wierzbicka puede verse en la Tabla 1.

El concepto de palabras «básicas» se debe a que se pretende definir todas las palabras a través de un conjunto muy restringido de los llamados primitivos semánticos [7]. La diferencia entre el vocabulario definidor y los primitivos semánticos es que las palabras del vocabulario definidor son las únicas palabras que pueden aparecer en las definiciones.

En cambio, los primitivos semánticos son independientes: no se puede definir unas a través de otras. Lo que significa que su conjunto es mínimo: no se puede remover de él ninguna palabra (primitiva semántica) sin perder la posibilidad de definir todas las demás palabras en el diccionario a través de este conjunto. Representando el diccionario como un grafo dirigido [8], la diferencia es que las palabras del vocabulario definidor deben ser accesibles en exactamente un paso por los vínculos del grafo, mientras que las primitivas semánticas pueden ser accesibles en varios pasos. Eso se debe al hecho de que las palabras del vocabulario definidor están presentes físicamente en las definiciones de las palabras (por eso el nombre de vocabulario definidor); a diferencia de los primitivos semánticos que se presentan virtualmente en las definiciones, por el hecho de ser accesibles en el grafo pasando, tal vez, por varios nodos.

Algunas observaciones reconocidas por la autora que se deben considerar con respecto a la lista antes descrita son [7]:

- 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 frasesmas, 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*).

**Tabla 1. Primitivas semánticas [6]**

Sustantivos	yo, tú, alguien, gente, algo, cuerpo
Determinantes	esto, lo mismo y otro
Cuantificadores	uno, dos, algo, todo mucho/muy
Evaluadores	bueno, malo
Descriptores	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, quizás, porque, si
Intensificador, aumentador	todo, más
Taxonomía, "partonomía"	tipo de, parte de
Similaridad	como

Apresjan [21] sugiere usar un vocabulario restringido que se considera la base para construir las definiciones del resto de las voces de una lengua, pero afirma que éste no puede ser tan pequeño como el marcado por Wierzbicka.

## 2.5 Teoría de grafos

La teoría de grafos, que nació como una rama de la Topología, hoy en día se ha convertido en una herramienta indispensable en diversos campos como: la investigación operativa, la lingüística, la química, la física, la genética, la teoría de redes o la teoría de la decisión.

La Teoría de Grafos es una herramienta indispensable para la fundamentación matemática de las Ciencias de la Computación. Los grafos permiten modelar fenómenos discretos y son indispensables en el manejo de estructuras de datos y el análisis de algoritmos.

En las ciencias computacionales, los grafos son utilizados en diferentes campos, tales como el procesamiento geoespacial, el diseño de circuitos electrónicos, la búsqueda de caminos mínimos y rutas óptimas, diseño de redes de computadoras etc.

El PLN ha utilizado los grafos como herramienta para solucionar diferentes tipos de problemas, como la creación de redes semánticas y la representación de diccionarios como es nuestro caso.

### 2.5.1. Conceptos de teoría de grafos

Se denomina grafo  $G$  al par  $(V(G), E(G))$ , en el que  $V(G)$  es un conjunto no vacío de elementos denominados vértices (también llamados nodos o puntos) y  $E(G)$  es un conjunto finito de pares no ordenados de elementos de  $V(G)$  denominados aristas. Al conjunto de elementos de  $V(G)$  se le denomina conjunto de vértices y al conjunto de elementos de  $E(G)$  conjunto de aristas.

Así, sea el conjunto de vértices  $V(G)=\{u, v, w, z\}$  y el conjunto de aristas  $E(G)=\{\{u, v\}, \{u, w\}, \{w, z\}\}$ , se dice que  $\{u, v\}$  es la arista que une los puntos  $u$  y  $v$ , y se le designa de

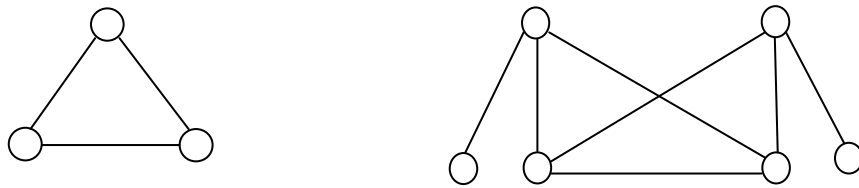


forma abreviada por  $uv$ . Si  $u=v$  la arista recibe el nombre de lazo. La representación de dicho grafo es la siguiente:

Se denomina grafo simple al grafo  $G=\{V(G), E(G)\}$  que verifica que para todo  $u, v$  perteneciente a  $V(G)$ , existe a lo sumo una única arista  $\{u, v\}$  de  $E(G)$  que los une.

$$\text{Grafo simple} \Leftrightarrow [\forall u, v \in V(G) \Rightarrow \exists! \{u, v\} \in E(G)]$$

Ejemplos de grafos simples serían los siguientes:



**Ilustración 3. Grafos simples**

Se dice que dos vértices  $u$  y  $w$  de un grafo  $G$  son adyacentes si el grafo contiene una arista que los une. Se dice también que los vértices son incidentes en dicha arista. Se dice que dos aristas son adyacentes si tienen, al menos, un vértice en común.

Se denomina grado de un vértice  $v$  de  $G$  al número de aristas que inciden en  $v$ , y se designará  $g(v)$  (un lazo en  $v$  contribuye de manera doble al grado de  $v$ ). A un vértice de grado 0 se le denomina vértice aislado. A un vértice de grado 1 se le denomina vértice terminal o extremo.

Un subgrafo de un grafo  $G$ , es un grafo cuyos vértices pertenecen a  $V(G)$  y cuyas aristas pertenecen a  $E(G)$ .

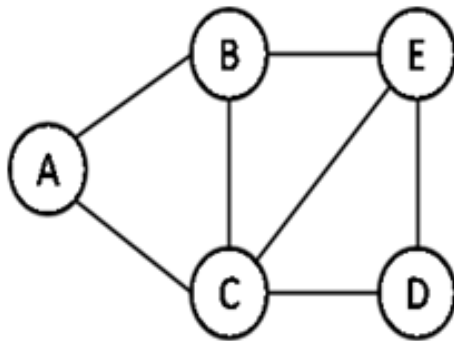
Un grafo es un conjunto de vértices  $V$ , más un conjunto de elementos  $F$  que conectan pares de vértices distintos. El grafo se describe por:  $G(V, F)$ .

Un grafo dirigido  $G$ , es una tupla  $G = (V, F)$  donde  $V \neq \emptyset$ , cuyos elementos denominamos vértices,  $F = V \times V$ , son los elementos de  $F$  denominados aristas dirigidas.

Dado un grafo  $G$  con  $m$  vértices y  $n$  aristas, podemos asociar a  $G$  una matriz de adyacencia. La matriz de adyacencia  $A=(a_{ij})$  de orden  $m \times m$  definida por:

$$a_{ij} = \begin{cases} k & \text{si } v_i \text{ es adyacente a } v_j \\ 0 & \text{en caso contrario} \end{cases}$$

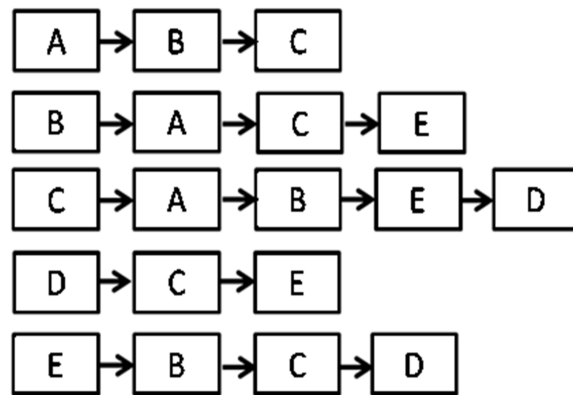
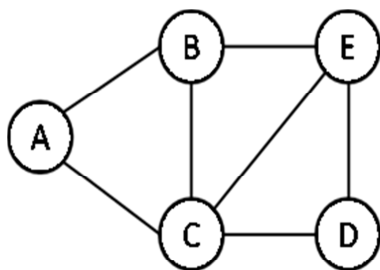
donde  $k$  es el número de aristas que unen el vértice  $v_i$  con el  $v_j$ , ver Ilustración 4.



	A	B	C	D	E
A	0	1	1	0	0
B	0	0	1	0	1
C	0	1	0	1	1
D	0	0	1	0	1
E	0	1	1	1	0

**Ilustración 4. Matriz de adyacencia**

Los grafos también pueden representarse usando una lista de adyacencia que consiste de un detalle de los vértices del grafo asociado a una lista que enumera sus vértices vecinos. Un ejemplo de una lista de adyacencia puede verse en la Ilustración 5.



**Ilustración 5. Lista de adyacencia**

Si en un grafo simple se recorren sucesivamente sus aristas de modo tal que dos sucesivas sean adyacentes, es decir que concurren al mismo vértice por el que se pasa de una a la otra, se está recorriendo o determinando una trayectoria o camino. Cuando cierta

trayectoria comienza y termina en el mismo nodo decimos que es un circuito o trayectoria cerrada, ver Ilustración 6.

Una trayectoria dirigida en  $G$  es una sucesión finita de vértices de  $G$  denotado como:

$$V_1, V_2, \dots, V_n$$

Diremos que una trayectoria dirigida es cerrada si y sólo si  $V_1 = V_n$ . Llamaremos ciclo dirigido a una trayectoria dirigida cerrada. Para nuestro problema se considera primitiva semántica al vértice  $V$  que cierra la trayectoria dirigida.

Sea  $G = (V, F)$  es un grafo dirigido, entonces  $G_I = (V_I, F_I)$  es un subgrafo de  $G$  si

$$V_I \neq \emptyset \text{ y } F_I \subset F,$$

Donde toda arista de  $F_I$  es incidente con los vértices de  $V_I$ .

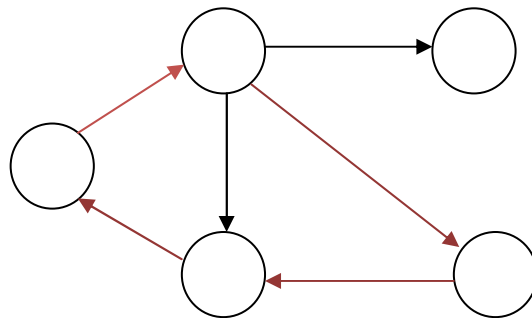


Ilustración 6 Trayectoria dirigida

## 2.6 Computación evolutiva

Existen problemas que por su complejidad no pueden resolverse usando algoritmos comunes. Para muchos de estos problemas ni siquiera podemos demostrar que tienen una solución óptima. Las técnicas clásicas de búsqueda y optimización son insuficientes para resolver problemas con espacios de búsqueda tan grandes. Las “heurísticas” son la mejor solución para este tipo de problemas. La palabra “heurística” se deriva del griego *heuriskein*, que significa “encontrar” o “descubrir”.

- 1 Inicio
- 2 Inicializar la población con soluciones candidatas aleatorias
- 3 Evaluar cada candidato
- 4 Repetir hasta (condición de terminación sea alcanzada) hacer
  - 5 ⇒ Seleccionar padres
  - 6 ⇒ Recombinar parejas de padres
  - 7 ⇒ Mutar los hijos resultantes
  - 8 ⇒ Evaluar los nuevos candidatos
  - 9 ⇒ Seleccionar los individuos de la siguiente generación
- 10 Fin de repetir
- 11 Salida: El mejor individuo de la población final
- 12 Fin

**Ilustración 7. Algoritmo evolutivo básico**

Las heurísticas fueron un área importante en los orígenes de la Inteligencia Artificial. Actualmente, el término suele usarse como un adjetivo, refiriéndose a cualquier técnica que mejore el desempeño en promedio de la solución de un problema, aunque no mejore necesariamente el desempeño en el peor caso [10].

Una heurística es una técnica que busca soluciones buenas aunque no pueden considerarse óptimas a un costo computacional razonable. En algunos casos, ni siquiera puede determinar qué tan cerca del óptimo se encuentra una solución, ver Ilustración 7 [11].

El término computación evolutiva o algoritmos evolutivos, representa una serie de técnicas inspiradas en los principios de la teoría Neo-Darwiniana de la evolución natural. En términos generales, para simular el proceso evolutivo en una computadora se requiere:

- Codificar las estructuras que se replicarán.
- Operaciones que afecten a los “individuos”.
- Una función de aptitud.
- Un mecanismo de selección.

**Tabla 2. Terminología utilizada en computación evolutiva**

<b>Termino</b>	<b>Concepto computacional [9]</b>
<b>Cromosoma</b>	Estructura de datos que contiene una cadena de parámetros de diseño o genes. Esta estructura de datos puede almacenarse, por ejemplo, como una cadena de bits o un arreglo de enteros.
<b>Gene</b>	Es una subsección de un cromosoma que (usualmente) codifica el valor de un solo parámetro.
<b>Genotipo</b>	Codificación (por ejemplo, binaria) de los parámetros que representan una solución del problema a resolverse.
<b>Fenotipo</b>	Es la decodificación del cromosoma. Es decir, los valores obtenidos al pasar de la representación (binaria) a la usada por la función objetivo.
<b>Aptitud</b>	Valor que se asigna a cada individuo y que indica que tan bueno es este con respecto a los demás.
<b>Generación</b>	Cada iteración de la medida de aptitud y a la creación de una nueva población por medio de operadores de reproducción.
<b>Operador genético o de reproducción</b>	Es aquel mecanismo que influencia la forma en que se pasa la información genética de padres a hijos.
<b>Cruza</b>	Operador que forma un nuevo cromosoma combinando partes de cada uno de sus cromosomas padres.
<b>Mutación</b>	Operador que forma un nuevo cromosoma a través de alteraciones (usualmente pequeñas) de los valores de los genes de un solo cromosoma padre.
<b>Elitismo</b>	Mecanismo utilizado para asegurar que los cromosomas de los miembros más aptos de una población se pasen a la siguiente generación sin ser alterados por los operadores genéticos

Existen muchas variantes de los algoritmos evolutivos (AE), aunque todas ellas mantienen un comportamiento básico: dada una población de individuos que representan un conjunto de posibles soluciones del problema, se genera un proceso de selección natural (la

supervivencia del más apto), misma permite el incremento en la aptitud de la población. Dada una función de calidad (*fitness function*) de maximización o minimización según sea el caso, se crean aleatoriamente un conjunto de soluciones que son evaluadas. Basados sobre esta aptitud, los mejores candidatos (aunque depende del proceso de selección) son seleccionados para pasar a la siguiente generación y aplicarles un método de recombinación y/o mutación. La recombinación es un operador que se aplica a dos o más candidatos seleccionados (padres) y resulta en uno o más nuevos candidatos (hijos).

La mutación es aplicada a un candidato y resulta un nuevo candidato. Realizando la recombinación y la mutación se obtiene un conjunto de nuevos candidatos (hijos) que son ahora los que compiten. Este proceso es iterativo hasta que algún criterio de finalización establecido sea alcanzado. Los operadores de variación (recombinación y mutación), crean la diversidad necesaria y consecuentemente exploración del campo de búsqueda. La selección, impulsa la calidad de las posibles soluciones.

El campo de aplicación de los AE es muy amplio, se pueden implementar para solucionar muchos de los problemas de la vida cotidiana y diversos problemas y modelos en ingeniería, por ejemplo:

- Los problemas de optimización: fueron la fuente de inspiración de los AE. Los AE se han utilizado tareas de optimización numérica, y los problemas de optimización combinatoria.
- Programación automática: Los AE se han empleado para desarrollar programas para tareas específicas, y para diseñar otras estructuras computacionales tales como los autómatas celulares y redes de clasificación.
- Aprendizaje máquina: Los AE se han utilizado también en muchas de estas aplicaciones, tales como la predicción del tiempo o la estructura de una proteína. Han servido asimismo para desarrollar determinados aspectos de sistemas particulares de aprendizaje, como pueda ser el de los pesos en una red neuronal, las reglas para sistemas de clasificación de aprendizaje o sistemas de producción simbólica, y los sensores para robots.

- Economía: En este caso, se ha hecho uso de estos algoritmos para modelar procesos de innovación, el desarrollo estrategias de puja, y la aparición de mercados económicos.
- Sistemas inmunes: A la hora de modelar varios aspectos de los sistemas inmunes naturales, incluyendo la mutación somática durante la vida de un individuo y el descubrimiento de familias de genes múltiples en tiempo evolutivo, ha resultado útil el empleo de esta técnica.
- Ecología: En la modelización de fenómenos ecológicos tales como las carreras de armamento biológico, la coevolución de parásito-huésped, la simbiosis, y el flujo de recursos.
- Genética de poblaciones: En el estudio de preguntas del tipo "¿Bajo qué condiciones será viable evolutivamente un gene para la recombinación?".
- Evolución y aprendizaje: Los AE se han utilizado en el estudio de las relaciones entre el aprendizaje individual y la evolución de la especie.
- Sistemas sociales: En el estudio de aspectos evolutivos de los sistemas sociales, tales como la evolución del comportamiento social en colonias de insectos, y la evolución de la cooperación y la comunicación en sistemas multiagentes.

Se dice que existen tres paradigmas principales:

- Programación Evolutiva,
- Estrategias Evolutivas,
- Algoritmos Genéticos.

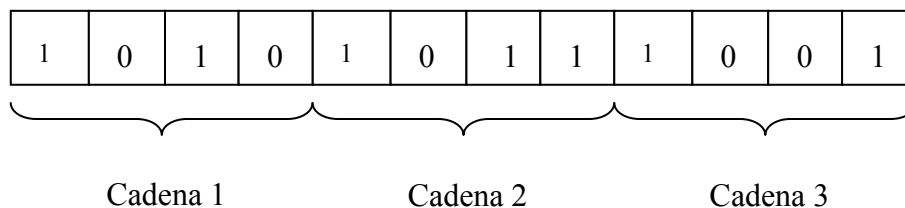
### **2.6.1 Algoritmo genético**

Los algoritmos genéticos llamados inicialmente "Planes reproductivos genéticos" fueron desarrollados por John H. Holland en la década de 1960, para resolver problemas de aprendizaje de máquina.

El algoritmo genético enfatiza la importancia de la cruce sexual (operador principal) sobre el de la mutación (operador secundario), y usa selección probabilística [10]. El algoritmo básico es el siguiente:

1. Generar (aleatoriamente) una población inicial.
2. Calcular aptitud de cada individuo.
3. Seleccionar (probabilísticamente) en base a aptitud.
4. Aplicar operadores genéticos (cruza y mutación) para generar la siguiente población.
5. Ciclar hasta que cierta condición se satisfaga.

La representación tradicional es la binaria, como puede verse en la Ilustración 8.



**Ilustración 8. Representación de la población**

A la cadena binaria se le llama “cromosoma”. A cada posición de la cadena se le denomina “gene” y al valor dentro de esta posición se le llama “alelo”. Para poder aplicar el algoritmo genético se requiere de los 5 componentes básicos siguientes:

- Una representación de las soluciones potenciales del problema.
- Una forma de crear una población inicial de posibles soluciones (normalmente es un proceso aleatorio).
- Una función de evaluación que juegue el papel del ambiente, clasificando las soluciones en términos de su “aptitud”.
- Operadores genéticos que alteren la composición de los hijos que se producirán para las siguientes generaciones.



- Valores para los diferentes parámetros que utiliza el algoritmo genético (tamaño de la población, probabilidad de cruce, probabilidad de mutación, número máximo de generaciones, etc.)

Algunas aplicaciones de los Algoritmos Genéticos son las siguientes: [39]

- Optimización (estructural, de topologías, numérica, combinatoria, etc.)
- Aprendizaje de máquina (sistemas clasificadores)
- Bases de datos (optimización de consultas)
- Reconocimiento de patrones (por ejemplo, imágenes)
- Generación de gramáticas (regulares, libres de contexto, etc.)
- Planeación de movimientos de robots
- Predicción

### **3.6.1.2 Representación por permutaciones**

Es utilizada en problemas de optimización combinatoria donde se busca una permutación que optimice una cierta función objetivo. Ejemplo.

Individuo1 = 9 8 4 5 6 7 1 2 3 10

Individuo2 = 8 7 1 2 3 10 9 5 4 6

Los operadores tradicionales no pueden ser aplicados trivialmente, es necesario utilizar operadores específicos.

### **2.6.1.3 Operador de selección**

El operador de selección es el encargado de transmitir y conservar las características de las soluciones que se consideran valiosas a lo largo de las generaciones. El principal medio para que la información útil se transmita es que los individuos mejor adaptados (mejor valor de función de evaluación) tengan más probabilidades de reproducirse. Sin embargo, es necesario también incluir un factor aleatorio que permita reproducirse a individuos que aunque no estén muy bien adaptados, puedan contener alguna información útil para

posteriores generaciones, con el objeto de mantener así también una cierta diversidad en cada población.

Uno de los métodos más utilizados es el de la Ruleta o Selección Proporcional: Con este método la probabilidad que tiene un individuo de reproducirse es proporcional a su valor de función de evaluación, es decir, a su adaptación. En este método se define un rango con las características de selección por sorteo. El número al azar será un número aleatorio forzosamente menor que el tamaño del rango. El elemento escogido será aquel en cuyo rango esté el número resultante de sumar el número aleatorio con el resultado total que sirvió para escoger el elemento anterior. El comportamiento es similar al de una ruleta, donde se define un avance cada tirada a partir de la posición actual. Lo anterior no permite escoger dos veces consecutivas el mismo elemento, y que puede ser forzado a que sea alta la probabilidad de que no sean elementos próximos en la población [38].

Existe también el método de Selección por Ranking: Desarrollado por Whitley (1989) consiste en calcular las probabilidades de reproducción de acuerdo con la ordenación de la población por el valor de adaptación y no simplemente por su valor de adecuación. Estas probabilidades se pueden calcular de diversas formas, aunque el método habitual es el ranking lineal (Baker, 1985).

El método de Selección por Torneo: tiene un valor computacional muy bajo dada su sencillez. Se selecciona un grupo de  $t$  individuos (normalmente  $t = 2$ ,  $t$ , torneo binario) y se genera un número aleatorio entre 0 y 1. Si este número es menor que un cierto umbral  $K$  (usualmente 0,75), se selecciona para reproducirse al individuo con mejor adaptación, y si este número es mayor que  $K$ , se selecciona, por el contrario, al individuo con peor adaptación. Esta técnica permite cierto grado de elitismo ya que el mejor nunca va a morir, y los mejores tienen más probabilidad de reproducirse. No produce convergencia prematura en el campo de búsqueda si la población es, al menos, un orden de magnitud superior al del número de elementos involucrados en el torneo.

#### **2.6.1.4 Operador de cruce**

Una vez seleccionados los individuos, éstos son recombinados para producir la población de la siguiente generación. Existen básicamente dos técnicas con las que puede operar el

factor de la cruce: Una estrategia destructiva, donde los descendientes se insertarán en la población temporal aunque sus padres tengan una mejor adaptación o una estrategia donde únicamente los hijos que superen la adaptación de sus padres pasaran a la siguiente generación.

El operador de cruce permite realizar una exploración de toda la información almacenada hasta el momento en la población y combinarla para crear mejores individuos.

Existen diferentes metodologías para la implementación de un algoritmo genético de permutaciones, para la realización de la cruce se contemplan las siguientes:

- *Order Crossover*: Propuesta por Davis (1985).
- *Partially Mapped Crossover (PMX)*: Propuesta por Goldberg y Lingle (1985). Tiene ciertas similitudes con OX.
- *Position-based Crossover*: Propuesta por Syswerda (1991) como una adaptación de la cruce uniforme para permutaciones.
- *Order-Based Crossover*: Propuesta por Syswerda (1991) como una ligera variante de *Position-based Crossover* en la que se cambia el orden de los pasos del algoritmo.
- *Cycle Crossover (CS)*: Propuesta por Oliver, Smith y Holland (1987). Es similar a la *Position-based Crossover*, porque toma algunos valores de un padre y selecciona los restantes del otro. La principal diferencia es que los valores tomados del primer padre no se seleccionan al azar, sino de acuerdo a ciertas reglas específicas.

### **2.6.1.5 Reemplazo de la población y condición de parada**

Cada vez que se aplica el operador de cruce, nos encontramos con un número de nuevos individuos (la descendencia) que se han de integrar en la población para formar la siguiente generación. Esta operación se puede hacer de diversas formas, pero en general existen tres métodos fundamentales para realizar el reemplazo:

- Se elimina un subconjunto de la población que contiene a los individuos peor adaptados para no rebasar un tamaño máximo de la población.

- Por cada nuevo individuo generado en la población se elimina el peor adaptado. Cada vez que se crea un nuevo individuo, en la población se elimina aleatoriamente una solución, independientemente de su adaptación.

El criterio de parada, generalmente es determinado por criterios como:

- Número máximo de generaciones
- Tiempo máximo de resolución,

Aunque en algunas ocasiones se utilizan estrategias que muestran con indicadores del estado de la población, como por la pérdida de diversidad dentro de la población o por no haber mejora en un cierto número de iteraciones. Una condición mixta es lo más utilizado, es decir, limitar el tiempo de ejecución a un número de iteraciones y tener en cuenta algún indicador del estado de la población para considerar la convergencia antes de alcanzar tal limitación.

Los Algoritmos Genéticos buscan identificar dentro de un espacio de búsqueda candidato, la mejor solución, donde la mejor solución es aquella que optimiza una medida numérica predefinida para el problema, llamada adaptación (*fitness*) de la solución.

El algoritmo opera iterativamente, actualizando un conjunto de posibles soluciones llamada población. En cada iteración, todos los miembros de la población son evaluados de acuerdo a una función de adaptación. Cada generación es creada seleccionando probabilísticamente los individuos de mayor adaptación en la nueva población. Algunos de estos individuos pasan intactos a la siguiente generación.

Otros son seleccionados para crear una nueva generación, aplicando operaciones genéticas como el cruce y la mutación.

Las entradas de este algoritmo incluyen una función de adaptación para evaluar los candidatos a posibles soluciones, un umbral que define el nivel aceptado de adaptación para dar por terminado el algoritmo, el tamaño que debe mantener la población, y los parámetros necesarios para determinar la fracción de la población que será remplazada en cada generación y la tasa de mutación presente.

Cada iteración de este algoritmo produce una nueva generación de posibles soluciones, con base en la población actual.

Primero, un cierto número de opciones en la población  $((1-r) \times p)$  es seleccionado probabilísticamente para su inclusión. La probabilidad de una de estas opciones  $h_i \in pob$  de ser seleccionada está dada por:

$$P_r(h_i) = \frac{Adaptación(h_i)}{\sum_{j=1}^p Adaptación(h_j)}$$

Por lo tanto, la probabilidad de que una posible solución sea seleccionada es proporcional a su propio índice de adaptación, e inversamente proporcional a la adaptación del resto de las soluciones en la misma población [38].

Una vez seleccionados las secciones que serán incluidos en la generación, el resto de los miembros de ésta se genera usando el operador de cruce. Este operador, selecciona  $(r \times p)/2$  pares de soluciones en la población y genera dos descendientes para cada par, combinando porciones de ambos padres. Los padres son elegidos probabilísticamente  $P_r(h_i)$ .

La mutación se considera un operador que proporciona de aleatoriedad en el entorno de los individuos de la población. Aunque el operador de cruce es el responsable de efectuar la búsqueda a lo largo del espacio de posibles soluciones, la mutación va ganando importancia a medida que la población de individuos va convergiendo. El objetivo del operador de mutación es producir nuevas soluciones a partir de la modificación de un cierto número de genes de una solución existente, con la intención de fomentar la variabilidad dentro de la población.

Para realizar el proceso de mutación a una población representada con permutaciones existen las siguientes alternativas:

- Mutación por inserción: selecciona un valor en forma aleatoria y se le inserta en una posición arbitraria.
- Mutación por Desplazamiento. Es una generalización de la mutación por inserción en la que en vez de mover un solo valor, se cambian de lugares varios a la vez.
- Mutación por Intercambio Recíproco. En este caso, se seleccionan dos puntos al azar y se intercambian estos valores de posición.

- Mutación Heurística. Fue propuesta por Cheng y Gen (1994). El procedimiento es el siguiente:
  - 1) Seleccionar genes al azar.
  - 2) Generar vecinos de acuerdo a todas las permutaciones posibles de los genes seleccionados.
  - 3) Evaluar todos los vecinos y seleccionar el mejor.

Sabemos que existen diferentes problemáticas para las que los Algoritmos genéticos podrían representar una posible solución. Lo que es cierto es que en toda ejecución de un Algoritmo Genético hay que decidir qué tipos de operadores son los que se adecuan a la situación en particular y con qué frecuencia es que éstos deben ser utilizados en la población. La mutación y otros operadores que generen diversidad se suelen aplicar con poca frecuencia; la recombinación se suele aplicar con frecuencia alta.

#### **2.4.1. Algoritmo de evolución diferencial**

La evolución diferencial (*ED*) es un algoritmo estocástico de optimización perteneciente a la categoría de computación evolutiva. Este algoritmo fue creado con el fin de simular procesos de evolución natural. Al igual que otros algoritmos de esta categoría, la *ED* posee una población de individuos (en este caso arreglos de permutaciones), los cuales se mezclan y mutan para producir nuevos individuos, los cuales serán elegidos de acuerdo a su desempeño [12]. La característica principal de la *ED* es el uso de vectores de prueba, los cuales compiten con los individuos de la población actual a fin de sobrevivir [11]. Este algoritmo se propone para resolver problemas de optimización, especialmente en espacios de búsqueda continua.

La *ED* aplica la mutación modificando al azar parte de la información de los individuos, permitiendo alcanzar zonas no exploradas del espacio de búsqueda. De este modo, las direcciones de búsqueda dependen de la localización de los individuos seleccionados para calcular los valores de mutación.

Tabla 3. Variaciones de ED

Nomenclatura	Variante
DE/rand/p/bin	$U_{i,j} = \begin{cases} x_{r_3,j} + F \cdot \sum_{k=1}^p (x_{r_1^k} - x_{r_2^k}) & \text{Si } U_j(0,1) < CR \text{ ó } j = j_r \\ x_{i,j} & \text{En otro caso} \end{cases}$
DE/rand/p/exp	$U_{i,j} = \begin{cases} x_{r_3,j} + F \cdot \sum_{k=1}^p (x_{r_1^k} - x_{r_2^k}) & \text{Si } U_j(0,1) < CR \text{ ó } j = j_r \\ x_{i,j} & \text{En otro caso} \end{cases}$
DE/best/p/bin	$U_{i,j} = \begin{cases} x_{r_{best},j} + F \cdot \sum_{k=1}^p (x_{r_1^k} - x_{r_2^k}) & \text{Si } U_j(0,1) < CR \text{ ó } j = j_r \\ x_{i,j} & \text{En otro caso} \end{cases}$
DE/best/p/bin	$U_{i,j} = \begin{cases} x_{r_{best},j} + F \cdot \sum_{k=1}^p (x_{r_1^k} - x_{r_2^k}) & \text{Si } U_j(0,1) < CR \text{ ó } j = j_r \\ x_{i,j} & \text{En otro caso} \end{cases}$
DE/current-to-rand/p	$\vec{u}_i = \vec{x}_i + K \cdot (\vec{x}_{r_3} - \vec{x}_i) + F \cdot \sum_{K=1}^p (\vec{x}_{r_1^p} - \vec{x}_{r_2^p})$
DE/current-to-best/p	$\vec{u}_i = \vec{x}_i + K \cdot (\vec{x}_{r_{best}} - \vec{x}_i) + F \cdot \sum_{K=1}^p (\vec{x}_{r_1^p} - \vec{x}_{r_2^p})$
DE/current-to-rand/p/bin	$U_{i,j} = \begin{cases} \vec{x}_{i,j} + K \cdot (\vec{x}_{r_3,j} - \vec{x}_{i,j}) + F \cdot \sum_{K=1}^p (\vec{x}_{r_1^p,j} - \vec{x}_{r_2^p,j}) & \text{Si } U_j(0,1) < CR \text{ ó } j = j_r \\ x_{i,j} & \text{En otro caso} \end{cases}$
DE/rand/2/dir	$\vec{v}_i = \vec{v}_1 + \frac{F}{2} (\vec{v}_1 - \vec{v}_2 + \vec{v}_3 - \vec{v}_4) \text{ donde } f(\vec{v}_1) < f(\vec{v}_2) \text{ y } f(\vec{v}_3) < f(\vec{v}_4)$

Ya que el algoritmo de evolución diferencial está enfocado a la mutación, permite ampliar la exploración en el campo de búsqueda evitando hasta cierto grado caer en mínimos locales. Es importante señalar que al aumentar el tamaño de la población o el número de pares de soluciones para calcular los valores de mutación, también aumentará la tendencia del algoritmo a explorar el espacio de búsqueda. Entonces, el equilibrio entre el tamaño de

la población y el número de mutaciones determinará la eficiencia del algoritmo [14]. El algoritmo general se muestra en la Ilustración 9.

Existe una nomenclatura desarrollada para hacer referencia a las diferentes variantes de evolución diferencial. La más popular es llamada “*DE/rand/1/bin*”, donde DE significa *Differential Evolution*, la palabra *rand* indica que se selecciona aleatoriamente los individuos para calcular los valores de mutación, “1” es el número de parejas de soluciones seleccionadas y “*bin*” significa que se utiliza recombinación binomial.

El parámetro llamado “CR”, controla la influencia del padre en la generación del hijo. Valores más altos implican menos influencia del padre. El parámetro “F” controla la influencia del conjunto de parejas de soluciones seleccionadas para calcular el valor de mutación. Incrementar el tamaño de la población o el número de parejas de soluciones para calcular los valores de mutación, también incrementará la diversidad de posibles movimientos, promoviendo la exploración del espacio de búsqueda, sin embargo, la probabilidad de encontrar la dirección de búsqueda correcta disminuye considerablemente. Por tanto, el balance entre el tamaño de población y el número de diferencias determina la eficiencia del algoritmo [13].

El valor de CR debe ser un número real entre [0,1] y el parámetro CR se recomienda que este entre [0.3,0.9] y se debe generar un valor nuevo en cada generación. En la Tabla 3 se muestran algunas variantes del algoritmo de evolución diferencial. Donde  $j_r$  es un número entero aleatorio generado de entre [1,D], siendo D el número de variables del problema.  $(0,1) U_j$  es un número real generado aleatoriamente entre 0 y 1. Ambos números se generan con una distribución uniforme [15].



### 2.4.2. Permutaciones con evolución diferencial

El algoritmo de evolución diferencial fue desarrollado originalmente para trabajar en optimización numérica global. Sin embargo, ha sido adaptado a otro tipo de problemas de optimización como la optimización combinatoria. La optimización combinatoria busca los mejores individuos que representan las posibles soluciones. En nuestro caso, los individuos

```
1 Inicio
2  G=0
3  Genera población inicial  $\vec{x}_{i,G} \forall i, i = 1, \dots, NP$ 
4  Evalúa  $f(\vec{x}_{i,G}) \forall i, i = 1, \dots, NP$ 
5  For G=1 to MAX GEN Do
6    For i=1 to NP Do
7      ⇒ Selecciona aleatoriamente  $r_1 \neq r_2 \neq r_3$ 
8      ⇒  $j_{rand} = randint(1,D)$ 
9      ⇒ For j=1 to D Do
10     ⇒ If ( $rand_j[0, 1) < CR$  or  $j = j_{rand}$ ) Then
11     ⇒  $u_{i,j,G+1} = x_{r_3,j,G} + F(x_{r_1,j,G} - x_{r_2,j,G})$ 
12     ⇒ Else
13     ⇒  $u_{i,j,G+1} = x_{i,j,G}$ 
14     ⇒ End If
15     ⇒ End For
16     If ( $f(\vec{u}_{i,G+1}) \leq f(\vec{x}_{i,G})$ ) Then
17        $\vec{x}_{i,G+1} = \vec{u}_{i,G+1}$ 
18     Else
19        $\vec{x}_{i,G+1} = \vec{x}_{i,G}$ 
20     End If
21   End For
22   G = G + 1
23 End For
24 End
```

Ilustración 9. Algoritmo de evolución diferencial

se representan por palabras del diccionario asociadas a un número entero. Así el algoritmo de ED representado en la Ilustración 9 se puede aplicar directamente a los vectores. Sin embargo, después de ejecutar el algoritmo los vectores resultantes están formados por números reales en vez de enteros, por lo que es necesario convertirlos a enteros nuevamente. Para ello, cada valor real del vector es sustituido por el número entero que correspondería a su posición si el vector fuera ordenado de forma ascendente [16].

Veamos un ejemplo:

Dados dos vectores de permutaciones  $x_{r1}$  y  $x_{r2}$ :

$$X_{r1} = \begin{bmatrix} 1 \\ 3 \\ 4 \\ 5 \\ 2 \end{bmatrix} \quad X_{r2} = \begin{bmatrix} 1 \\ 4 \\ 3 \\ 5 \\ 2 \end{bmatrix}$$

El subíndice,  $f$ , denota un vector de representación de coma flotante.

$$X_{r1,f} = \frac{x_{r1}}{5} = \begin{bmatrix} 0.2 \\ 0.6 \\ 0.8 \\ 1 \\ 0.4 \end{bmatrix} \quad X_{r2,f} = \frac{x_{r2}}{5} = \begin{bmatrix} 0.2 \\ 0.8 \\ 0.6 \\ 1 \\ 0.4 \end{bmatrix}$$

Se selecciona de forma aleatoria un tercer vector.

$$X_{r3} = \begin{bmatrix} 5 \\ 2 \\ 1 \\ 4 \\ 3 \end{bmatrix} \rightarrow X_{r3,f} = \begin{bmatrix} 1 \\ 0.4 \\ 0.2 \\ 0.8 \\ 0.6 \end{bmatrix}$$

Se aplica la mutación.

$$v_f = x_{r3,f} + F(x_{r1,f} - x_{r2,f})^{F=0.85} = \begin{bmatrix} 1 \\ 0.4 \\ 0.2 \\ 0.8 \\ 0.6 \end{bmatrix} + 0.85 \begin{bmatrix} 0 \\ -0.2 \\ 0.2 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 0.23 \\ 0.37 \\ 0.8 \\ 0.6 \end{bmatrix}$$

El vector mutante de coma flotante se transforma de nuevo en el dominio entero asignando el valor más pequeño flotante (0.23) al entero más pequeño (1), el siguiente valor más alto flotante (0.37) al entero inmediato superior (2) y, así sucesivamente, hasta obtener:

$$v_f = \begin{bmatrix} 1 \\ 0.23 \\ 0.37 \\ 0.8 \\ 0.6 \end{bmatrix} \rightarrow v = \begin{bmatrix} 5 \\ 1 \\ 2 \\ 4 \\ 3 \end{bmatrix}$$

Este proceso permite utilizar representación de permutaciones sin afectar el comportamiento general del algoritmo.

# Capítulo 3

## Estado del arte

### 3.1 LDOCE como red semántica

Kozima y Furugori (1993) [19] crearon una red semántica para el 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  $n$ . El nodo que representa a cualquier palabra tiene enlaces con todos los 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 una palabra aparece en la definición de otra u otras palabras, existirán enlaces desde los nodos de estas palabras hacia el nodo de ésta. Por ejemplo:

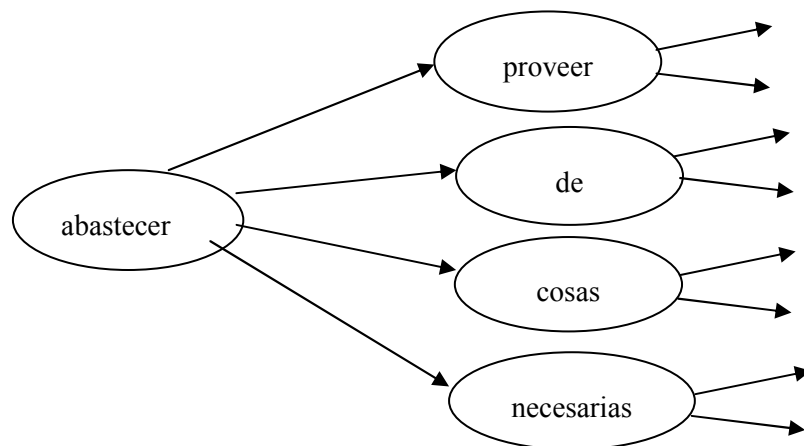


Ilustración 80. Nodos en la red semántica

La red semántica separó las palabras que pertenecen al LDV y lo llamaron “Glosema”. Éste conjunto 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 para esa palabra, en el LDOCE. Posteriormente el Glosema se tradujo subsecuentemente en una red semántica llamada "Paradigma". 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. Estas ligas son denominadas "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*".

La similitud entre palabras en el LDV se calcula a través de la función de activación "*spreading activation*". Todos los nodos tienen cierta "actividad", expresada en el campo "*activity-value*" o  $v_n$ , que es recibida y transmitida a través de las ligas de los nodos y se calcula como sigue:

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

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,
- $\theta$  es una función que limita la salida al intervalo  $[0,1]$ .

El algoritmo que calcula la similitud  $sim_{KF}(w_k, w_l)$  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  $P(w_k)$ . La similitud es:

$$sim_{KF}(w_k, w_l) = s(w_l) \cdot a(P(w_k), w_l)$$

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

Donde:

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

$N$  es el número de palabras del corpus.

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

Para que abarque todo LDOCE se aplica la siguiente fórmula:

$$sim_{KF}(w, w') = \Psi\left(\sum_{w' \in W'} s(w') \cdot a(P(W), w')\right)$$

Donde:

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

El método de Kozima y Furugori, es un método que mide la distancia de las palabras sin tomar en consideración la influencia del contexto. Kozima e Ito crearon un método más dinámico y sensible al contexto; la medida que utilizan toma en consideración la "dirección asociativa" de un par dado de palabras

## 3.2 Extracción automática de primitivas semánticas con métodos aleatorios

El diccionario fue representado como un grafo dirigido donde, los vértices son las palabras que se mencionan en el diccionario, tanto las entradas, como las que se usan en las definiciones. La misma palabra puede ocurrir tanto como una entrada como varias veces en las definiciones, contándose como el mismo vértice, ver Ilustración 11. Se aplicaron dos formas diferentes para la representación del diccionario: por forma base y 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.

El grafo dirigido fue construido palabra por palabra evitando en cualquier momento la existencia de ciclos en las definiciones. El orden de entrada de las palabras fue dado por diferentes métodos aleatorios como: método aleatorio uniforme, método por frecuencias y método por votación aleatoria.

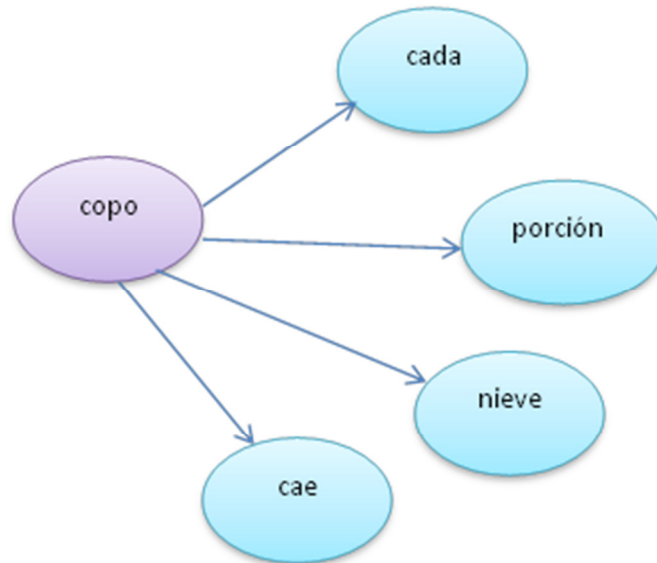
Método 1. Genera un número aleatorio entre los  $N$  nodos de la red, de manera que cada uno de los nodos sea evaluado como candidato a ser primitiva.

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.

Construyeron con el diccionario tradicional dos tipos distintos de grafos (por lexemas y por significados), probaron diferentes ordenamientos para obtener el menor conjunto definidor posible:



**Ilustración 91. Representación del diccionario**

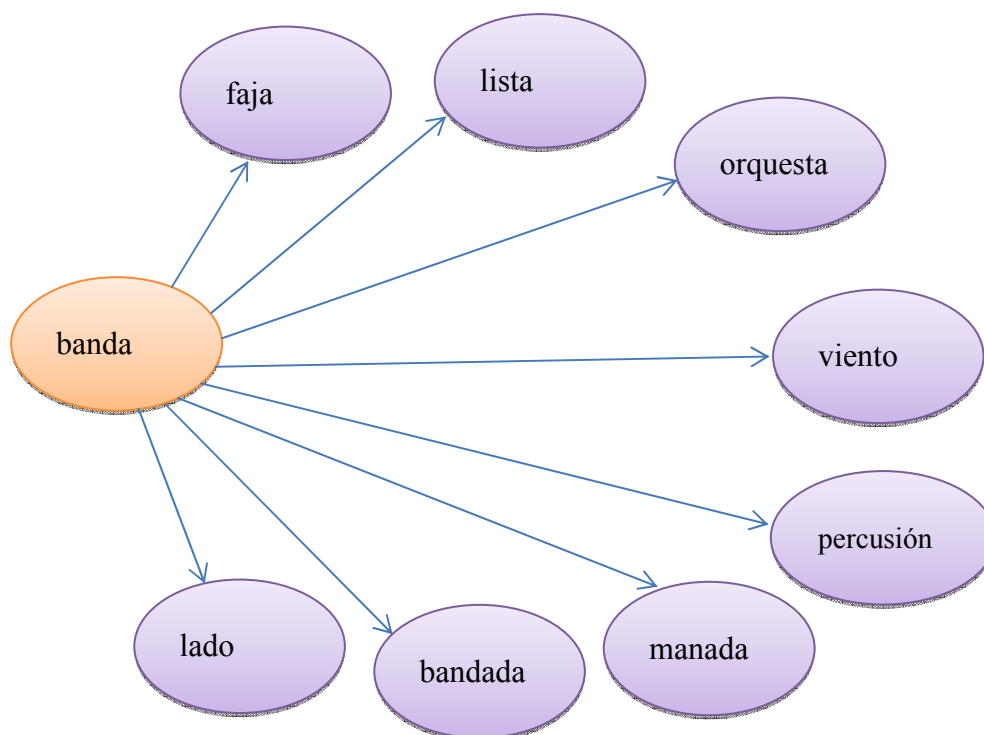
Método 1: aleatorio, uniforme. Usaron el ordenamiento uniformemente aleatorio: en cada iteración del algoritmo, eligieron aleatoriamente el siguiente vértice  $\sigma(i)$  de los vértices todavía no procesados.

Método 2: por frecuencias. Ordenaron 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  $G'$  y los de mayor frecuencia tendieron a ser definidores y entrar en  $P$ . Se esperaba con esta heurística,  $P$  sería menor porque los vértices que lo forman rompen más ciclos en  $G$ .

Método 3: aleatorio, por frecuencias. Este método es una combinación de los métodos 1 y 2. Usaron el ordenamiento aleatorio, pero con las probabilidades en función inversa a las frecuencias: en cada iteración del algoritmo  $A$ , se eligió 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,  $\sum p_i = 1$ .



Método 4: por votación aleatoria. En este método, generaron 20 diferentes conjuntos definidores mínimos  $P_i$  con el método 1, y para cada vértice, contaron el número de los conjuntos  $P_i$  en los cuáles éste entra, asociando cada vértice con un número entre 20 (entró en todos) y 0 (nunca entró). Ordenaron 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, desde 1 a 20.



**Ilustración 102. Grafo tipo 1.**

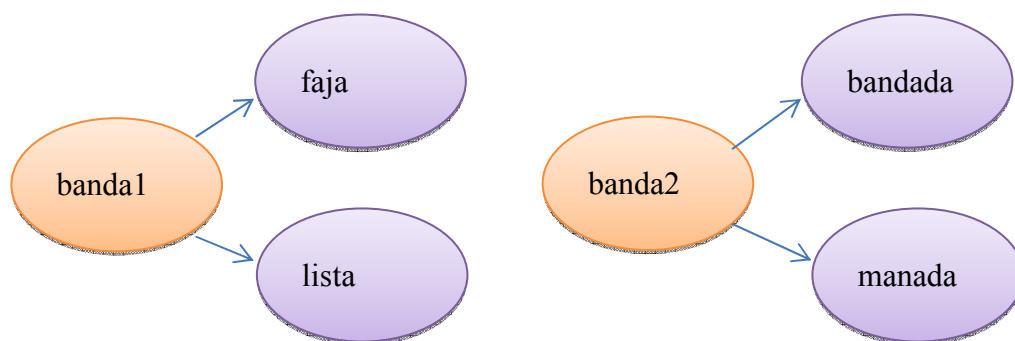
Grafo 1: por lexemas. En este experimento, consideraron un lexema como un vértice del grafo, es decir, una palabra normalizada morfológicamente. En este método, las definiciones de todos los significados de la palabra y de todas las entradas homónimas– se consideraron como una larga definición. – Ilustración 12.

Grafo 2: por significados. En este experimento, consideraron un vértice del grafo como un significado específico de la palabra, por ejemplo: gato1a (animales), gato1b (tipo de animales), gato2a (herramienta) etc. Donde para desambiguar los significados de las palabras que forman las definiciones, emplearon un etiquetador (*tagger*) para la

normalización morfológica y para la desambiguación de la parte de oración, y un algoritmo parecido al de Lesk. En ambos casos, no se consideran las preposiciones, conjunciones, verbos auxiliares, etc., ver Ilustración 13.

Para los experimentos utilizaron el Diccionario de la lengua española del grupo Anaya (1997). Este diccionario contiene 30,971 entradas, de los cuales sólo 30,725 son palabras significativas. Estas palabras significativas se dividen en 60,818 significados específicos [14].

El método por votación aleatoria obtuvo el mejor resultado con un total de 2,246 primitivas semánticas, número que los autores consideran significativo dada su cercana coincidencia con el tamaño del conjunto de lemas existentes en el *LDOCE* y, como una consideración adicional, es un número comparable con el total de los ideogramas que conforman el vocabulario chino básico. Este trabajo fue el primero que trató de obtener un conjunto de primitivas automáticamente. Su enfoque obtuvo un conjunto de 2,246 palabras que podría constituir el conjunto de primitivas para español. Aunque se debe considerar una posible dispersión en los datos dado el carácter aleatorio de los métodos utilizados.



**Ilustración 113. Grafo tipo 2**

# Capítulo 4

## Método propuesto

Se propone obtener un conjunto de primitivas semánticas tomando como fuente de información dos diccionarios explicativos para español. Ambos diccionarios serán preprocesados y se generará a partir de ellos una lista que enumere las palabras que sirven como entradas al diccionario y las palabras contenidas en sus definiciones. El grafo será construido palabra por palabra utilizando algoritmos evolutivos que buscarán la mejor permutación de entrada utilizando las palabras contenidas en la lista previamente desarrollada. El esquema computacional se muestra en la Ilustración 14.

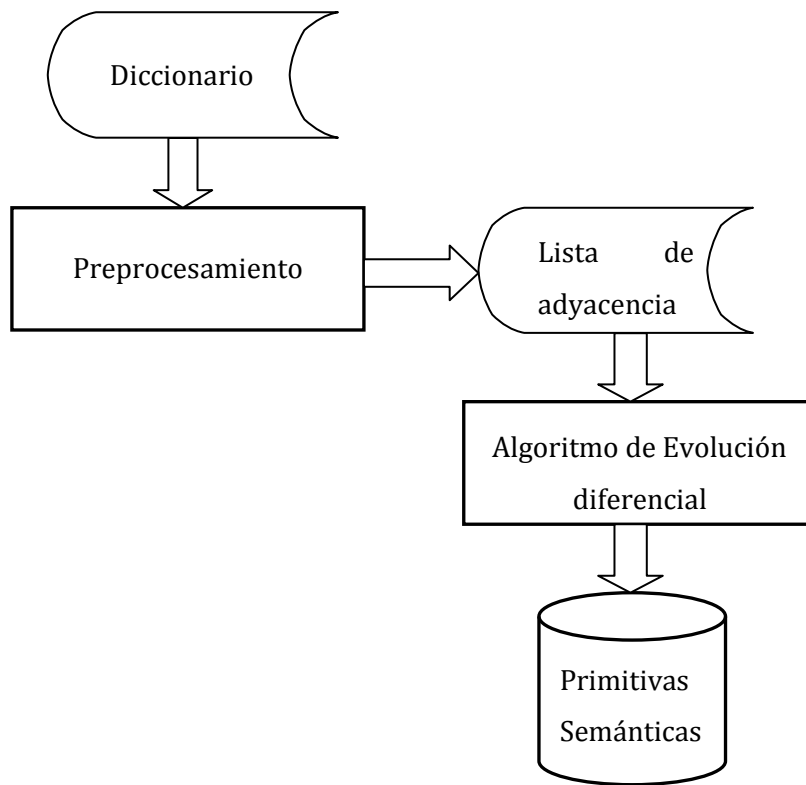
La búsqueda de la permutación de entrada que permita la obtención de un conjunto de primitivas semánticas mínimo es considerado de tipo NP completo dado que El número de permutaciones para el diccionario Anaya es del orden de  $10,011!$  y para el diccionario de RAE de  $15,219!$ . Por lo anterior este problema no puede ser resuelto con un cálculo directo y la aplicación de algún método heurístico se hace indispensable.

Debido a las características del problema se decidió la implementación de algoritmos evolutivos que aunque no garantizan el óptimo de los resultados si pueden proporcionar una buena solución. Se utilizaron dos algoritmos: Algoritmo Genético y Algoritmo de Evolución Diferencial que generarán diferentes permutaciones de entrada para la definición del grafo.

Para la construcción del grafo se retomó la técnica utilizada por Rivera-Loza *et al* [8]. Construyendo el grafo correspondiente a cada diccionario palabra por palabra verificando después de cada inserción la existencia de un nuevo ciclo.

Se utilizaron dos diccionarios explicativos para español:

- Diccionario de la lengua española del grupo Anaya, edición 1997,
- Diccionario de la Real Academia Español (RAE) edición 2007.



**Ilustración 124. Modelo computacional**

## 4.1 Selección de diccionarios

Los diccionarios son herramientas lingüísticas que recogen el léxico de una lengua. Existen diferentes tipos de diccionarios para fines de esta investigación, nos concentraremos en los diccionarios monolingües, generales y explicativos.

Este tipo de diccionarios son llamados “pasivos” dado que únicamente están orientados a la comprensión de voces y oraciones, y no a la generación.

Las secciones textuales dispuestas ordenadamente en un diccionario son denominadas artículos, y se conforman por una entrada también denominada unidad léxica, y la información que la define. Las entradas pueden ser simples (una sola palabra) o

complejas (más de una palabra), y aparecen ordenadas alfabéticamente en el diccionario en su forma lematizada.

La definición constituye el elemento central del artículo lexicográfico, por lo que es necesario conocer el tipo de información que ésta maneja. Desde esta perspectiva, debemos primero subrayar que el tipo de definición que a nosotros interesa, la definición lexicográfica, da únicamente información sobre el contenido lingüístico de la unidad léxica.

Las unidades léxicas pueden dividirse en: palabras de contenido léxico (sustantivos, adjetivos, verbos y adverbios) y palabras funcionales (preposiciones, pronombres, etc.). De acuerdo a lo anterior existirán dos tipos de definiciones: la definición propia o perifrástica y definición impropia o funcional.

La definición propia expresa el *significado* de las entradas en cuanto a su contenido léxico-semántico, es decir, se utiliza con las palabras de contenido léxico. La definición impropia se utiliza para *explicar* el funcionamiento y empleo de palabras funcionales, dado que carece de un verdadero significado léxico

La estructura de las definiciones propias suele seguir la norma establecida por la llamada definición aristotélica, la cual consiste de un enunciado encabezado por un término genérico o hiperónimo inmediato, seguido de una diferencia específica, o conjunto de rasgos y características que diferencian el término definido de otros que se agrupan bajo el mismo hiperónimo [17].

El diccionario explicativo de mayor importancia en países hispanohablantes es el Diccionario de la Real Academia Española (DRAE), el cual también es utilizado como punto de referencia de otros diccionarios generales. es por esto que su edición de 2007 será uno de los diccionarios que utilizaremos en la investigación nuestra investigación. Este diccionario cuenta con 70,000 entradas y cerca de 120,000 definiciones.

El segundo diccionario a utilizar es el Diccionario de la lengua española del grupo Anaya, edición 1997. Se utiliza este diccionario por su limitado número de entradas (alrededor de 30,000) y por la alta frecuencia con que hace uso de ciclos cortos en las definiciones.

Se espera poder comparar el comportamiento de los diferentes algoritmos con diccionarios de diferente número de entradas y diferente calidad en las definiciones.

## 4.2 Preprocesamiento de los diccionarios

Para poder utilizar el diccionario como entrada en un algoritmo evolutivo es indispensable transformarlo para que la información ahí contenida sea más accesible. Se busca además reducir en lo posible el costo computacional de las operaciones porque reducimos en lo posible el tamaño del grafo. Esta tarea fue dividida en siete pasos: eliminar de información adicional, eliminar de prefijos y sufijos, eliminar de entradas contenidas en su propia definición, etiquetado morfológico y eliminar *stop words* y eliminar entradas no utilizadas en el resto de las definiciones.

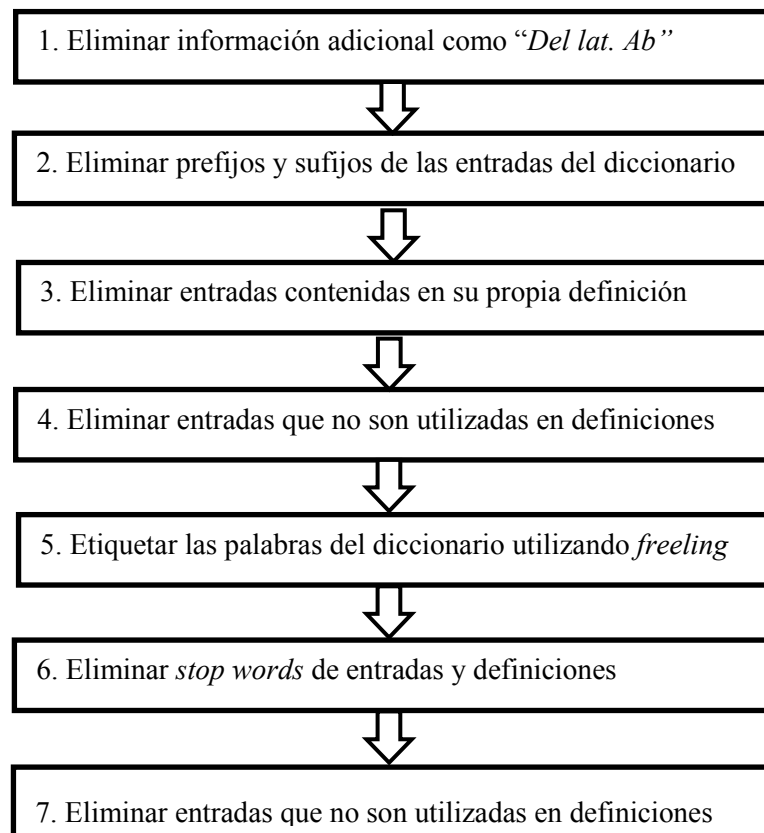


Ilustración 135. Preprocesamiento del diccionario

### 5.1.2. Eliminar información adicional

La definición en las unidades léxicas no incluye únicamente la explicación de la entrada, en la mayoría de los casos también se incluye información como el origen etimológico o cuestiones morfológicas, por ejemplo:

abecedario

1. **Del lat. abecedarium (sustantivo masculino).**  
Serie ordenada de las letras de un idioma.
2. **(sustantivo masculino).**  
Cartel o librito para enseñar a leer.
3. **(sustantivo masculino).**  
Conjunto de signos que se emplean para usar un lenguaje:
  - manual
  - para los sordos, telegráfico, etc.

Ilustración 16. Ejemplo. Información adicional en la definición

En el ejemplo de la Ilustración 16 la unidad léxica “*abecedario*” incluye en su definición el origen etimológico de la palabra, información morfológica para dos de los sentidos e indica una palabra sinónima. Toda esta información habrá de ser retirada porque únicamente trabajaremos con las explicaciones de cada una de las entradas.

### 5.1.2. Eliminar prefijos y sufijos en las definiciones

Los sufijos y prefijos forman parte del proceso de derivación, que permite crear nuevos vocablos a partir de otros ya existentes. El sufijo se agrega hacia el final de las palabras, mientras que el prefijo se ubica al comienzo.

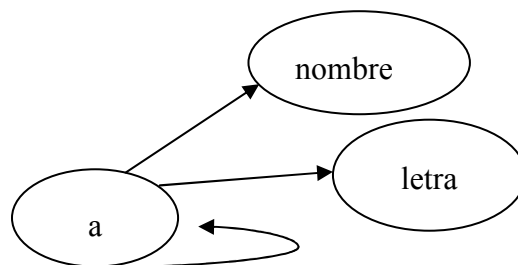
a-, an-  
E: Elementos que entran en la formación de  
diversas palabras

**Ilustración 147. Ejemplo de prefijo**

Aunque los prefijos y sufijos forman parte de las entradas del diccionario no fueron incluidos en la representación porque no son palabras, son elementos afijos carentes de autonomía que necesariamente tienen que unirse a una base léxica a la que aportan valores creando nuevas palabras. Ilustración 17.

#### **4.1.2. Eliminar de las entradas contenidas en su propia definición**

Con el fin de reducir en lo posible el número de palabras en el grafo se eliminaron las entradas que estaban contenidas en su propia definición porque al ser incluidas en el grafo formaban un lazo. Al ser éstas palabras que cierran un ciclo fueron consideradas como primitivas semánticas. Ilustración 18.



**Ilustración 158. Palabra que genera lazo**



#### 4.1.2. Eliminar entradas que no son utilizadas en otras definiciones

Existe un grupo de palabras que están contenidas como unidades léxicas en el diccionario pero que no son utilizadas en las definiciones de otras entradas del diccionario, puesto que estas palabras nunca estarían en posibilidad de cerrar un ciclo en el grafo fueron eliminadas de la lista de entradas para reducir el número de palabras en el grafo y reducir así el costo computacional de la creación del grafo.

Este es un proceso iterativo dado que cada vez que elimina una entrada del diccionario se elimina también su definición, permitiendo que existan nuevas palabras que ya no son utilizadas en las definiciones. Un ejemplo puede verse en la Ilustración 19.

```
Eliminar "pescada" en la primera iteración

    pescada
        merluza. cecial, pescado seco.

Permite que "cecial" no sea utilizada en ninguna
definición.
    cecial
        merluza u otro pescado parecido seco y
curado
```

Ilustración19. Palabras sin uso en las definiciones

#### 4.1.2. Etiquetado morfológico y eliminación de palabras gramaticales

Las palabras existentes en una lengua se dividen en dos grandes grupos: *palabras plenas* (o *palabras autónomas*, *palabras autosemánticas*, *palabras lexicales*, etc.), y *palabras gramaticales* (o bien *palabras auxiliares*, *palabras sinsemánticas*, *palabras vacías*, etc.). Para la primera clase, cada palabra designa por sí mismas un concepto léxico autónomo, es decir, hacen referencia al mundo real o abstracto, indicando personas, objetos, acciones, estados, ideas, características, propiedades, etc. Las palabras plenas se clasifican en sustantivos, verbos, adjetivos y adverbios.

En la segunda clase agrupa las palabras que no tienen significado léxico, y que se utilizan para establecer relaciones que se dan entre las palabras que ocurren de manera en el enunciado expresado (relaciones sintagmáticas). Estas palabras no pueden existir de manera autónoma en una oración, siempre deben ir acompañadas por al menos una palabra plena. Dentro de esta clase de palabras encontramos las conjunciones, las preposiciones, los numerales y los artículos.

*FreeLing* es una biblioteca que proporciona servicios de análisis del lenguaje está diseñado para ser utilizado como una biblioteca externa pero puede ser utilizada con su propia interfaz desde la línea de comandos. Ésta herramienta proporciona servicios como: análisis morfológico, clasificación de entidades nombradas, etiquetación (*PoS tagging*), etc. Los idiomas soportados actualmente son: español, catalán, gallego, italiano, inglés, ruso, portugués, galés y asturiano con diferentes servicios para cada idioma.

Con el fin de normalizar el diccionario manejando los lemas de las palabras y de utilizar exclusivamente *palabras plenas*: verbos, adverbios, sustantivos y adjetivos se etiquetó todo el diccionario utilizando *Freeling* [18].

Una vez etiquetadas fueron eliminadas las *palabras gramaticales* de las definiciones. Dada la eliminación de palabras de las definiciones es necesario repetir el paso cuatro (eliminar de palabras que no se usan en otras definiciones). Este proceso es iterativo ya que cada eliminación hace necesario una nueva búsqueda de palabras no utilizadas.

Una vez generada la lista de entradas, se asignó un número entero a cada una de ellas, el cual se usa como índice. Este índice identificará a la palabra y permitirá al algoritmo evolutivo trabajar únicamente con números y no con la cadena de caracteres.

### 4.3 Aplicación de los algoritmos evolutivos

Dado  $\sigma$ , que es una permutación de entrada generada por un algoritmo de evolución diferencial, el grafo  $G_I$  se construye, palabra por palabra. Inicialmente,  $G_I$  está vacío. Se construye insertando uno a uno los vértices y sus relaciones hacia los vértices ya insertados. Con cada inserción en  $G_I$  se verifica que no se genere un ciclo en las definiciones, de ser

así, el vértice no se inserta y se le considera una primitiva semántica. El algoritmo propuesto obtiene, dado un grafo dirigido  $G = (V, F)$ , un subconjunto definidor  $P$  donde  $p \subseteq V$  es considerada una primitiva semántica, si cualquier ciclo en el grafo  $G$  contiene un vértice de  $P$ . Llamaremos a los vértices  $p \in P$  primitivas semánticas. [8]

Sabemos que el orden en que las palabras son ingresadas al grafo es relevante, por lo que usaremos una heurística que busque las mejores permutaciones de entrada para las palabras del diccionario. Se utilizaron dos heurísticas: Algoritmo Genético y Algoritmo de Evolución Diferencial.

Ambos algoritmos, requieren como entrada una lista de índices de palabras para generar los vectores de enteros permutados que formarán la población inicial.

La representación de los individuos será con permutaciones. Esto es, cada palabra tiene asignado un índice., El algoritmo de Evolución Diferencial genera diferentes permutaciones de los índices asociados a las entradas del diccionario:

$$x_i^m = [1 \ 2 \ 3 \ 4 \ 5 \ \dots \ n]$$

donde  $n$  es el total de entradas en el diccionario y  $m$  el total de vectores en la población.

Las heurísticas tendrán como entrada la lista de índices de las palabras que conforman el diccionario, generará diferentes vectores de permutaciones y creará los grafos que correspondan a cada una de éstas, buscando la minimización del conjunto de primitivas.

El valor de aptitud de un individuo en la población es el valor numérico obtenido después de evaluar la función objetivo en la población que definió de la siguiente forma:

$$\textit{Minimizar } |P|, \textit{ donde } P\{x \mid x \in V \square x \notin G1\}$$

$|P|$  es el tamaño del conjunto de primitivas más pequeño obtenido después de varias ejecuciones del algoritmo.

Después del preprocesamiento el diccionario Anaya mantuvo un total de 10,011 entradas y el diccionario de la Real Academia Española 15,219 Se creó una matriz de adyacencia para cada diccionario que sirvió como entrada a los algoritmos evolutivos.

## 4.1 Configuración del algoritmo genético

Para la implementación del algoritmo genético se requiere la configuración de un conjunto de parámetros como se muestra.

### 4.1.1 Método de selección

El algoritmo propuesto utilizó el método de selección por ruleta de De Jong. Éste algoritmo es el más utilizado en la configuración de Algoritmos Genéticos.

Para este método se le asigna una parte proporcional de la ruleta a cada uno de los individuos de la ruleta, de tal forma que la suma de todos los porcentajes sea la unidad. Los mejores individuos recibirán una porción de la ruleta mayor que la recibida por los peores. Generalmente la población está ordenada en base al ajuste por lo que las porciones más grandes se encuentran al inicio de la ruleta. Para seleccionar un individuo basta con generar un número aleatorio del intervalo  $[0..1]$  y devolver el individuo situado en esa posición de la ruleta. Esta posición se suele obtener recorriendo los individuos de la población y acumulando sus proporciones de ruleta hasta que la suma exceda el valor obtenido.

Es un método muy sencillo, pero ineficiente a medida que aumenta el tamaño de la población (su complejidad es  $O(n^2)$ ). Presenta además el inconveniente de que el peor individuo puede ser seleccionado más de una vez. También se le conoce como selección de Montecarlo.

### 4.1.2 Cruza

La idea principal de la cruce es que al seleccionar individuos correctamente adaptados y cruzarlos podríamos obtener un individuo que herede el buen comportamiento de sus padres. Para nuestro algoritmo se utilizó una cruce ordenada, este tipo de cruce implica poca complejidad algorítmica lo que permitió reducir en lo posible el tiempo de procesamiento del algoritmo. La cruce ordenada se describe a continuación.

Cruce ordenada, ejemplo:

Individuo1 = [9 8 4 5 6 7 1 2 3 10]

Individuo2 = [8 7 1 2 3 10 9 5 4 6]

Sub-cadena elegida: 5 6 7 1 (de P1) Primer hijo:

Individuo1 = X X X 5 6 7 1 X X X

Borrar de P2 la sub-cadena tomada de P1:

Individuo2\_0 = 8 X X 2 3 10 9 X 4 X

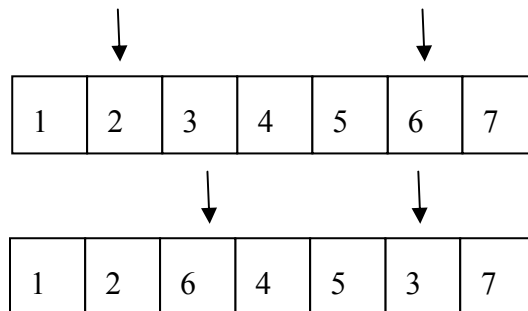
Determinar los valores faltantes de H1 sustituyendo (de izquierda a derecha) los valores que aparecen en P2':

IR = 8 2 3 5 6 7 1 10 9 4

Para obtener IR, el procedimiento es similar, aunque ahora la sub-cadena se tomará de P2 y la sustitución se hará a partir de P1'.

### 4.1.3 Mutación por intercambio recíproco

La mutación por intercambio recíproco selecciona dos posiciones aleatorias y luego intercambia las ciudades sobre estas posiciones.



El número de ejecuciones para un algoritmo evolutivo recomendado es de 30. Para nuestro experimento el algoritmo fue ejecutado en 38 ocasiones con diferentes configuraciones en los parámetros, como puede verse en la Tabla 4.

Tabla 4. Configuración de parámetros del Algoritmo Genético

	Número de individuos	Número de generaciones	Elitismo	Porcentaje de mutación	Resultados
Ejecución 1	300	500	No	0.15	2,456
Ejecución 23	500	500	No	0.1	2,353
Ejecución 32	500	300	Yes	0.2	2,305
Ejecución 33	500	300	Yes	0.2	2,362

## 4.2 Configuración del algoritmo de evolución diferencial

Se implementó un algoritmo de ED, *DE/rand/p/bin* donde  $p=1$ .

$$U_{i,j} = \begin{cases} x_{r_3,j} + F \cdot \sum_{k=1}^p (x_{r_1^k} - x_{r_2^k}) & \text{Si } U_j(0,1) < CR \text{ ó } j = j_r \\ x_{i,j} & \text{En otro caso} \end{cases}$$

Donde la recombinación binomial es equivalente a decidir con cierta probabilidad, si la posición actual tomaría el valor del vector padre o el del vector de mutación, y se define como sigue:

$$U_{i,j,G} = \begin{cases} v_{i,j,G} & \text{si } (rand_j[0,1] \leq CR) \text{ o } (j = j_{rand}) \\ x_{i,j,g} & \text{en cualquier otro caso} \end{cases}$$

Donde  $CR$  es una constante de recombinación definida por el usuario en el rango de  $[0, 1]$  y  $j_{rand}$  es un entero seleccionado de manera aleatoria en el rango  $[1, NP]$  para garantizar que el vector hijo  $U_{i,G}$  sea diferente en al menos un parámetro del vector padre  $X_{i,G}$ .

El valor de  $CR$  debe ser un número real entre  $[0,1]$  y el parámetro  $CR$  se recomienda que este entre  $[0.3,0.9]$  y se debe generar un valor nuevo en cada generación.  $j_r$  es un número entero aleatorio generado de entre  $[1,D]$ , siendo  $D$  el numero de variables del problema.  $(0,1) U_j$  es un número real generado aleatoriamente entre 0 y 1. Ambos números

se generan con una distribución uniforme [11]. En la Tabla 3 se muestran algunas variantes del algoritmo de evolución diferencial

El número de ejecuciones para un algoritmo evolutivo recomendado por los expertos es de 30. Para nuestro experimento los algoritmos fueron ejecutados en 38 ocasiones con diferentes configuraciones en los parámetros, como puede verse en la Tablas 4 y 5.

**Tabla 5. Configuraciones de parámetros del Algoritmo de Evolución Diferencial**

	<b>Número de individuos</b>	<b>Número de generaciones</b>	<b>Elitismo</b>	<b>Porcentaje de mutación</b>	<b>Resultados</b>
Ejecución 1	300	500	No	0.15	2,263
Ejecución 23	500	500	No	0.1	2,259
Ejecución 32	500	300	Yes	0.2	2,169
Ejecución 33	500	300	Yes	0.2	2,163

### **4.3 Descripción de los experimentos**

La lista de las palabras que se usan para la detección de las primitivas semánticas se representa como una lista de adyacencia y tiene un total 10011 palabras, cada una de éstas con sus relaciones —palabras que participan en su definición—. Los algoritmos generaron diferentes permutaciones de esta lista que sirvieron como población inicial en ambos casos y que fueron recombinadas durante cada iteración de éste. Se generó el grafo para cada una de estas recombinaciones verificando la ausencia de ciclos.

Tanto el Algoritmo Genético como el Algoritmo de Evolución diferencial fueron configurados para utilizar elitismo en algunas ejecuciones, es decir, para que el mejor individuo de cada generación se mantuviera en la siguiente. El elitismo permite mantener los mejores resultados obtenidos en cada iteración pero restringe la variación.

Los dos algoritmos fueron ejecutados en 38 ocasiones ajustando en cada una la configuración de los parámetros.

Para nuestros experimentos , al igual que el generado por Rivera-Loza y otros (2003), los primeros vértices en entrar al grafo normalmente no se consideran por el algoritmo como primitivas en cambio los últimos elementos de la lista tienden a ser primitivas (entrar en  $P$ ). Dado que las palabras con lazos fueron excluidas desde el preprocesamiento, la palabra asociada al primer índice en entrar al grafo nunca es considerada como primitiva.

Se diseñaron cuatro tipos diferentes de experimentos. El primero consistió en reducir el tamaño del conjunto obtenido mediante el uso de algoritmos evolutivos:

- Algoritmo de evolución diferencial,
- Algoritmo genético.

El objetivo de este experimento es reducir en lo posible el número de primitivas generadas por anteriores investigaciones esto con el fin de reducir la cantidad de palabras que son excluidas del grafo.

Los siguientes experimentos consisten en evaluar el conjunto obtenido y analizar sus propiedades.

El *Longman Dictionary of Contemporary English* (LDOCE por sus siglas en inglés) (Longman, 1991) retoma el concepto de Apresjan utilizando lo que ellos denominan un vocabulario definidor (*defining vocabulary*). En este diccionario todas las definiciones son construidas usando exclusivamente el vocabulario definidor, que es un vocabulario con 2,206 lemas.

El LDOCE [20] fue el primer diccionario que estuvo disponible en un formato para computadora y es también el más usado actualmente. Este diccionario es particularmente especial porque posee un vocabulario definidor, el *Longman Defining Vocabulary* (LDV) de 2866 palabras, que se emplean para formar las 56,000 definiciones de palabras restantes. Este vocabulario fue creado utilizando como base la Lista de Servicios Generales de Michael West (*Michael West's General Service List*) que fue construida en 1953 y que está compuesta por las palabras con mayor frecuencia en un corpus de inglés escrito.

El segundo experimento está relacionado con la evaluación del conjunto. Se considera que el LDOCE al ser creado de forma manual y por un grupo de lingüistas expertos tiene un conjunto de palabras al que creemos de calidad. Nuestro experimento consiste en medir



la intersección existente entre el conjunto de palabras obtenidas por el algoritmo y el vocabulario del LDOCE.

Puesto que el vocabulario del LDOCE fue creado para inglés se generó una traducción al español. En esta traducción, los vocablos que tienen dos o más acepciones fueron agrupados como una bolsa de palabras. Existen palabras en inglés que tienen hasta diez acepciones diferentes en español lo que hizo que el tamaño real del conjunto de comparación creciera hasta: 8,145 palabras.

Una de los planteamientos Wierzbicka indica que el conjunto de primitivas semánticas debería ser universal, verificar la correlación entre estos conjuntos –uno creado para el español y otro para inglés- podría apoyar esta idea.

El tercer experimento consistió en medir la frecuencia del conjunto de primitivas en dos diferentes corpus:

- Se midió la frecuencia absoluta y relativa de la lista de palabras obtenidas en el propio diccionario.
- Se midió la frecuencia absoluta y relativa de la lista de palabras obtenidas en Internet.

La hipótesis detrás de esta evaluación es que las palabras más frecuentes son mejores candidatas a ser las primitivas semánticas. Una frecuencia alta en estos corpus mostrará la utilidad del conjunto en el uso habitual del vocabulario tanto por usuarios comunes de la red como por lexicógrafos encargados de la elaboración de diccionarios explicativos.

En el cuarto experimento, siguiendo la hipótesis de Kozima y Furugori [19], confirmaremos la densidad asociada a las palabras obtenidas, así entre mayor sea la frecuencia del conjunto en su totalidad se considerará mayor calidad del conjunto.

Los experimentos antes mencionados se realizaron con dos diccionarios: El Diccionario de la lengua española del grupo Anaya, edición 1997 y el Diccionario de la Real Academia Española (RAE) edición 2007.

# Capítulo 5

## Discusión de los resultados

Tanto el Algoritmo Genético (AG) como el Algoritmo de Evolución Diferencial fueron ejecutados para el diccionario Anaya (con 10,011 entradas) como para el diccionario de Real Academia Española RAE (con 15,219 palabras) en 38 ocasiones con diferentes configuraciones en los parámetros. La configuración que obtuvo mejores resultados en ambos diccionarios se muestra en la Tabla 6.

**Tabla 6. Configuraciones con mejores resultados.**

Algoritmo Genético
<ul style="list-style-type: none"><li>- 500 individuos.</li><li>- 300 generaciones.</li><li>- CR= 0.2 (el grado de recombinación entre individuos de la población).</li><li>- F= 0.5 (el valor de mutación de los vectores de prueba).</li></ul>
Algoritmo de Evolución Diferencial
<ul style="list-style-type: none"><li>- 500 individuos.</li><li>- 300 generaciones.</li><li>- CR= 0.2 (el grado de recombinación entre individuos de la población).</li><li>- F= 0.5 (el valor de mutación de los vectores de prueba).</li></ul>

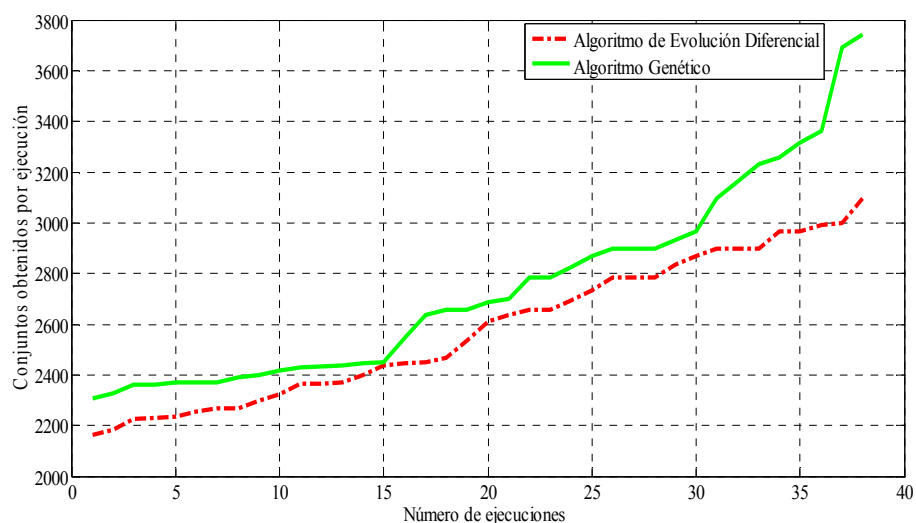
El mejor resultado del Algoritmo Genético creó un conjunto de 2,305 palabras (ya incluido el conjunto de palabras que generaban lazos), 59 palabras más que el conjunto generado por Rivera-Loza y otros (2003).

Los resultados obtenidos con el Algoritmo genético para el diccionario Anaya no mejoraron el tamaño del conjunto generado en otras investigaciones, sin embargo el conjunto que se obtuvo es de mayor calidad que el generado con métodos aleatorios. Con

mayor calidad nos referimos a que en promedio la frecuencia de uso del vocabulario obtenido por el Algoritmo genético es más elevada que la del conjunto de comparación.

En caso del diccionario de la RAE el algoritmo genético no obtuvo mejores resultados que el Algoritmo de Evolución diferencial. En un comparativo entre las diferentes ejecuciones el AG nunca estuvo por encima de los resultados proporcionados por el AED, como se muestra en la Ilustración 20.

Lo anterior nos lleva a concluir que el algoritmo genético tiende a concentrarse en óptimos locales con mayor frecuencia que el algoritmo de evolución diferencial. Lo anterior resalta la utilidad de algoritmos enfocados a la mutación para la solución de problemas de procesamiento de lenguaje natural.



**Ilustración 2016. Comparación de los resultados del AG y AED para diccionario Anaya**

Los conjuntos de primitivas fueron evaluados con respecto a su grado de intersección con la traducción del vocabulario del LDOCE. La lista de palabras de la traducción del LDOCE es de 8,145, puesto que muchas de las palabras tienen más de una acepción de español. Al cotejar la intersección de los conjuntos no se verificó la palabra original en inglés únicamente la existencia en la lista. El algoritmo generó un conjunto con 101 palabras más que el utilizado por Longman.

**Tabla 7. Resultados diccionario Anaya con Algoritmo Genético**

	<b>Número de Primitivas</b>	<b>Coincidencias con LDOCE</b>
Rivera-Loza y otros	2,246	1,596 (71.05% del conjunto)
Algoritmo Genético	2,305	1,489 (64.59% del conjunto)

En esta evaluación el conjunto Rivera- Loza y otros obtuvo mejores resultados que el AG tanto para el diccionario Anaya como para el diccionario de RAE ya que el número de coincidencias éste con respecto a la traducción fue mayor. En la Tabla 7 se muestran los resultados antes mencionados.

Aunque sólo el 64.59% del conjunto obtenido pertenece también al conjunto de LDOCE ambos conjuntos, tanto el Rivera-Loza como el generado por el algoritmo genético están cerca del número de lemas que define el vocabulario Longman y reducen en 561 palabras –el algoritmo genético– y en 620 –el método Rivera-Loza– el número establecido por el vocabulario LDOCE para el inglés aun cuando el conjunto utilizado para la comparación es una traducción al español.

Para el diccionario de la RAE el algoritmo genético obtuvo un conjunto de 3652 para un conjunto de entrada de 15,219 palabras en el grafo. Este número sobrepasa por mucho el menor obtenido por el método aleatorio, sin embargo se tiene que considerar la calidad –en nuestro caso un diccionario de mayor calidad aquel tiene menor cantidad de ciclos en sus definiciones– de ambos diccionarios y que el número de palabras entrantes es mayor.

Al medir el número de palabras obtenidas por el algoritmo genético se puede verificar que mantiene un comportamiento constante con respecto al número de palabras utilizadas como entradas en el grafo. Con el diccionario Anaya y un total de 10011 entradas se obtuvo un conjunto de 2,305 palabras lo que corresponde al 23.04% del total del vocabulario. Para

el diccionario de RAE se generó un conjunto de 3652 palabras que corresponde al 24.99% del conjunto de entrada.

La configuración para el Algoritmo de Evolución Diferencial obtuvo mejores resultados para el diccionario Anaya un conjunto de 2,163 (ya incluido el conjunto de palabras que generaban lazos), 83 palabras menos que el conjunto generado por Rivera-Loza y otros (2003).

**Tabla 8. Resultados diccionario Anaya con AED**

	<b>Número de Primitivas</b>	<b>de</b>	<b>Coincidencias con LDOCE</b>
Rivera-Loza y otros	2,246		1,596 (71.05% del conjunto)
Algoritmo de Evolución Diferencial	2,163		1,598 (73.87% del conjunto)

El número de palabras obtenidas corresponde al 21.60% del conjunto total de entrada al grafo. Para el caso del diccionario Anaya el algoritmo redujo aún más el número de palabras obtenidas, se obtuvo un conjunto con un número de palabras menor al conjunto de lemas definidos por LDOCE. En este caso el algoritmo de evolución diferencial obtuvo 43 palabras menos que el conjunto total lemas.

También se calculó el porcentaje de las coincidencias relativas entre el conjunto obtenido con el Algoritmo de Evolución diferencial y el LDOCE, que se obtiene dividiendo el número de primitivas en *LDOCE* entre el tamaño del conjunto de primitivas. El cálculo de este porcentaje muestra que cantidad de las primitivas coinciden con la lista *LDOCE* y puede observarse en la tabla 9.

Para el conjunto de primitivas obtenido con respecto a la lista traducida del LDOCE de 2,163 palabras pertenecientes al conjunto generado por el algoritmo de Evolución diferencial 1,598 coinciden con el conjunto LDOCE lo que equivale al 72.43% del total de

lemas del LDOCE. Esta medida muestra que porcentaje de *LDOCE* está cubierto por el conjunto obtenido.

**Tabla 9. Resultados para el Diccionario Anaya, Algoritmo de Evolución Diferencial**

	<b>Número de primitivas</b>	<b>Coincidencias absolutas con <i>LDOCE</i></b>	<b>Coincidencias relativas con <i>LDOCE</i></b>
Mejor resultado obtenido	2,163	1,598 (56.05% del conjunto)	73.87%
Promedio de los resultados obtenidos	2,580	1,655 (58.04% del conjunto)	64.14%

Es importante mencionar que el conjunto generado por el algoritmo de evolución diferencial es 2% más pequeño que el vocabulario creado de forma manual.

**Tabla 10. Resultados obtenidos para Anaya: comparación con el conjunto Rivera-Loza *et al.***

	<b>Número de primitivas</b>	<b>Coincidencias absolutas con <i>LDOCE</i></b>	<b>Coincidencias relativas con <i>LDOCE</i></b>
Algoritmo de ED	2,163	1,598 (56.05%)	73.87%
Rivera Loza y otros (2003)	2,246	1,487 (52.15% <sup>1</sup> )	66.2%

El Algoritmo genético generó un total de 3,165 primitivas para el diccionario de la Real Academia Española (20.79% del vocabulario original). El total de palabras obtenido por el Algoritmo de Evolución diferencial para el diccionario de RAE es de 3,043 entradas, lo que corresponde al 19.99% del vocabulario de entrada al grafo. Como puede observarse el porcentaje es muy similar para ambos diccionarios, ver Tabla 11.

**Tabla 11. Comparación entre AG y AED con RAE**

<b>Número de Primitivas</b>	<b>Coincidencias con <i>LDOCE</i></b>
-----------------------------	---------------------------------------

Algoritmo genético	3,165	2,114 (66.79% del conjunto)
Algoritmo de Evolución Diferencial	3,043	2,378 (78.11% del conjunto)

En la Tabla 10 presentamos comparación del conjunto obtenido por el algoritmo de evolución diferencial con el método aleatorio —conjunto generado por Rivera-Loza y otros (2003) —.

Tanto el Algoritmo genético como el Algoritmo de Evolución diferencial obtienen conjuntos que corresponden a un rango entre el 19% y el 21% de vocabulario de entrada de los diccionarios.

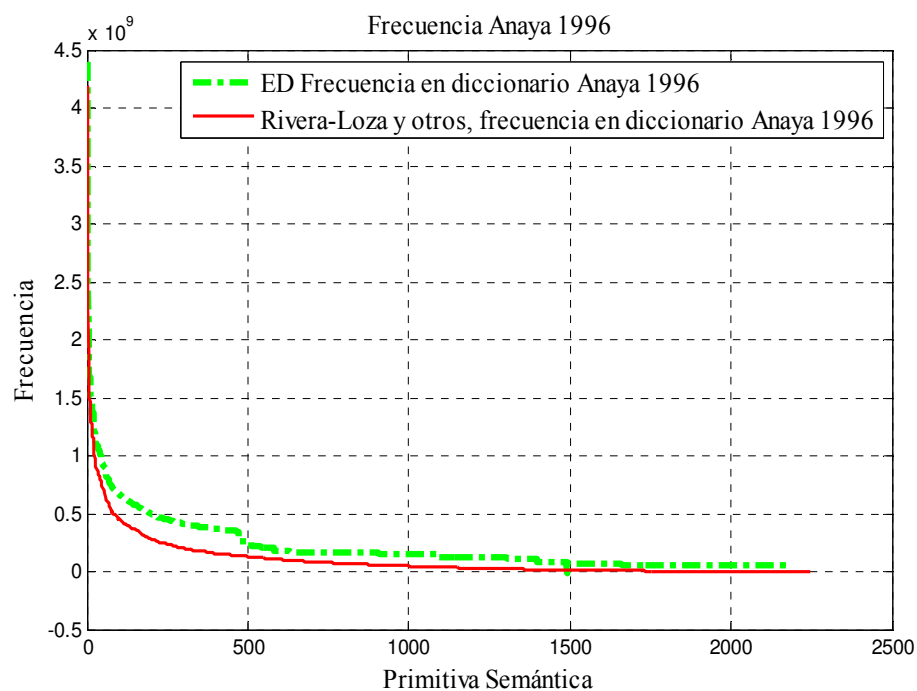
**Tabla 12. Resultados obtenidos para RAE: comparación con el conjunto Rivera-Loza**

	<b>Número de primitivas</b>	<b>Coincidencias absolutas con <i>LDOCE</i></b>	<b>Coincidencias relativas con <i>LDOCE</i></b>
Algoritmo de ED	3,043	2,037 (92.33%)	66.94%
Rivera Loza y otros	2,246	1,487 (52.15% <sup>1</sup> )	66.2%

La siguiente evaluación consistió en medir la frecuencia de palabras en Internet. Se realizaron consultas en Internet con cada una de las palabras de ambos conjuntos utilizando la herramienta *Yahoo!Search*. Los resultados se ordenaron por su frecuencia en Internet. Como puede verse en la Ilustración 22, el conjunto alcanzado con el algoritmo de evolución diferencial se mantiene siempre por encima del conjunto aleatorio, aunque la diferencia entre las listas no es muy grande. Es importante mencionar que la frecuencia que se muestra puede cambiar con respecto de un buscador a otro y de una fecha a otra —aunque el comportamiento tiende a aumentar en ambos casos— El eje X representa el rango de la primitiva semántica, el eje Y representa su frecuencia (en este caso hay que multiplicar por  $10^9$ ). Realizando el resto de los valores normalizados en cada punto de la gráfica y sumando los resultados obtenemos el valor de mejora de 4.8%.

Otra evaluación también toma en cuenta las frecuencias pero toma las frecuencias en el propio diccionario. La frecuencia obtenida para el diccionario Anaya, con un comportamiento similar al anterior el algoritmo de evolución diferencial obtuvo una mejora del 1.3% con respecto a Rivera-Loza y otros (2003) como se puede ver en la Ilustración 21.

Con respecto al promedio de los resultados obtenidos, se realizó una depuración del conjunto identificando cuales son las palabras que se repetían en los grupos obtenidos en diferentes corridas. Lo anterior arrojó un acumulado de 2,023 palabras que redundantes en dos o más conjuntos. El conjunto antes mencionado puede ser objeto de análisis y conformar con él la base de otro tipo de búsquedas ya que al verificar la frecuencia del conjunto en internet y el mismo diccionario obtienen los resultados más elevados y coinciden en su mayoría con la lista de primitivas arrojada por el algoritmo. Para completar el número calculado como media, se ordenaron el resto de las palabras de acuerdo su frecuencia en el mismo diccionario y se extrajeron las 557 palabras restantes. De las palabras establecidas para el conjunto promedio 1,655 coinciden con el conjunto LDOCE lo que corresponde al 58.04%.

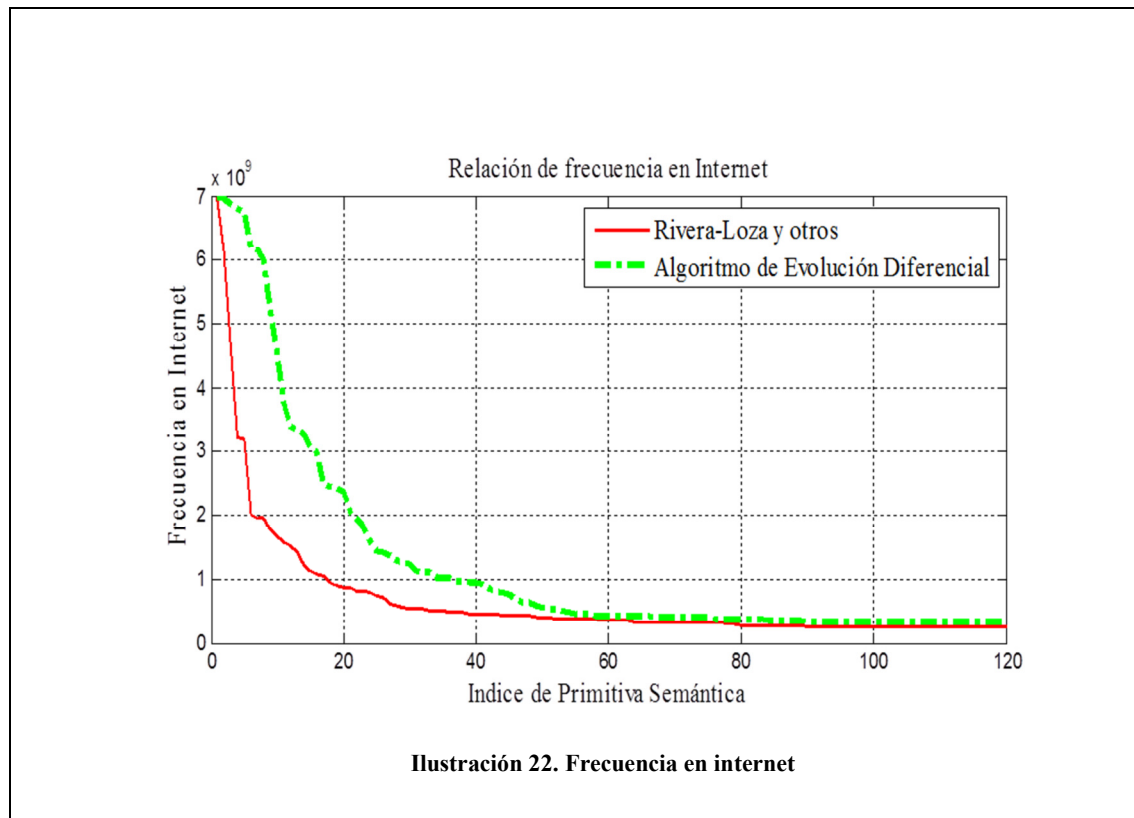


**Ilustración 21. Frecuencia en el diccionario Anaya**



Se puede concluir que comparando el total de las palabras ingresadas en el grafo, las palabras que tienden a entrar en el conjunto de primitivas semánticas son palabras con las frecuencias más elevadas tanto en Internet como en el mismo diccionario. Se obtienen los mejores resultados con el algoritmo de la evolución diferencial.

Es posible que el tamaño del conjunto óptimo oscile entre los 2,100 y 2,500 que es el rango del tamaño de los conjuntos obtenidos en diferentes ejecuciones lo que pudimos verificar es que entre mayor es la calidad del conjunto menor es su tamaño.



A continuación se muestran algunas de las palabras que coincidieron en los cuatro conjuntos generados por los algoritmos evolutivos y por el método aleatorio. Esta lista contiene parte de las palabras con más frecuencia de uso en Internet y el propio diccionario.

**Tabla 13. Primitivas semánticas con mayor frecuencia**

Total	4,830,000,000
Ver	3,460,000,000
Principal	1,350,000,000
Fin	1,320,000,000
Sólo	13,280,000,000
Vez	1,160,000,000
Tipo	1,610,000,000
Natural	2,410,000,000
País	1,210,000,000
Área	5,380,000,000
Base	2,390,000,000
Hotel	3,150,000,000
Sur	3,920,000,000
Ya	4,290,000,000
Día	2,570,000,000
Idea	1,810,000,000
Rock	2,990,000,000
Material	2,480,000,000
Social	4,380,000,000
Simple	2,820,000,000
Grande	2,010,000,000
General	5,150,000,000
Local	5,160,000,000
Plan	2,700,000,000
Final	2,810,000,000

Color	4,060,000,000
Para	8,240,000,000
Simple	2,820,000,000
Social	4,380,000,000
Error	2,700,000,000

# Capítulo 6

## Conclusiones y trabajo futuro

### 6.1 Conclusiones

Se implementaron algoritmos evolutivos para extraer un conjunto de palabras del diccionario, que al considerarse primitivas semánticas crean un diccionario sin ciclos.

Se implementó un algoritmo de evolución diferencial que mejoró los resultados obtenidos por trabajos anteriores en un 7.3% respecto al tamaño del conjunto obtenido para el diccionario Anaya en la edición 1997.

El algoritmo de evolución diferencial no sólo minimiza el conjunto de primitivas, también logra obtener un conjunto de palabras con mayor calidad, según los criterios establecidos.

Se obtuvieron conjuntos de palabras cuya frecuencia en el uso habitual es mayor que la que se obtuvo en otras investigaciones.

Verificamos que la frecuencia de uso de las palabras consideradas primitivas semánticas es relevante en el momento de su selección.

De acuerdo con los resultados obtenidos, los algoritmos evolutivos enfocados a la mutación pueden proporcionar mejores resultados aplicados en problemas de procesamiento de lenguaje natural.

El tamaño del conjunto de primitivas semánticas obtenidas con diferentes algoritmos y diferentes diccionarios representa en todos los casos cerca del 20% del lenguaje utilizado en la construcción del grafo.

## **6.2 Aportaciones**

Aportaciones científicas:

1. Método de extracción de primitivas semánticas de diccionarios explicativos con algoritmos bioinspirados que da mejores resultados que otros métodos.
2. Evaluación de los resultados de extracción de primitivas semánticas que muestra que los algoritmos bioinspirados dan mejores conjuntos de primitivas semánticas.

Aportaciones técnicas:

1. Conjunto de palabras consideradas primitivas semánticas para dos diccionarios explicativos para español.
2. Dos diccionarios explicativos para español sin ciclos con ausencia de vocablos.
3. Implementación de un algoritmo de evolución diferencial para la extracción de primitivas semánticas.
4. Implementación de un algoritmo genético diferencial para la extracción de primitivas semánticas.

## **6.3 Trabajo futuro**

Algunas de las direcciones para el trabajo futuro son:

1. Implementar otros algoritmos que permitan verificar los resultados obtenidos.
2. Verificar si otros diccionarios mantienen el mismo comportamiento que los utilizados en los experimentos.
3. Buscar alternativas de representación para las palabras excluidas de los diccionarios.
4. Generar un diccionario computacional sin ciclos en las definiciones.

# Referencias

- [1] Brodie, M. L. & Manola F. (1989) Database Management: A Survey, in J. W. Schmidt & C. Thanos (eds.).
- [2] Boguraev, B. (1991) Building a Lexicon: The Contribution of Computers, en B. Boguraev (ed.).
- [3] Boguraev, B. & Pustejovsky J. (1996) Corpus Processing for Lexical Acquisition. Cambridge, Mass: The MIT Press.
- [4] Moreno-Ortiz, A. (2000) Diseño e implementación de un lexicón computacional para lexicografía y traducción automática. Facultad de Filosofía y Letras. Universidad de Málaga. Esp. Volumen 9, ISSN: 1139-8736.
- [5] Goddard, C. (1999) Polysemy: a problem of definition. In Yael Ravin and Claudia Leacock (Eds). Polysemy: Theoretical and computational approaches. Oxford: Oxford University Press, pp.129-151.
- [6] Wierzbicka, A. (1980) *Lingua Mentalis: The semantics of natural language*. New York: Academic Press.
- [7] Wierzbicka, A. (1996) *Semantics: Primes and Universals*. Oxford: Oxford University Press.
- [8] Rivera-Loza, G., Gelbukh, A., Sidorov, G. (2003) Selección automática de primitivas semánticas para un diccionario explicativo del idioma español. Tesis de maestría, Laboratorio de Lenguaje Natural y Procesamiento de Texto, CIC, Instituto Politécnico Nacional. México.
- [9] Coello-Coello, C. (2011) *Introducción a la Computación Evolutiva (Notas de Curso)*. Departamento de Computación CINVESTAV-IPN. México
- [10] Coello-Coello, C. (2008) *Introducción a la computación evolutiva (Notas de Curso)*. CINVESTAV-IPN Departamento de Computación. México.

- [11] Reeves, C. B. (ed.) (1993) *Modern Heuristic Techniques for Combinatorial Problems*. John Wiley & Sons, Great Britain.
- [12] Mezura, E., Velázquez, J. y Coello, C. (2006). A comparative study of differential evolution variants for global optimization. In *Proc. of the 8th annual conference on GECCO, USA, 2006* pp. 485- 492.
- [13] Fogel, L. J., Owens, A. J. y Walsh, M. J. (1966) *Artificial Intelligence Through Simulated Evolution*, Wiley & Sons, Inc., New York.
- [14] León-Javier, A. (2009) *Diseño e implementación en hardware de un Algoritmo bioinspirado*. Centro de Investigación en Computación. Instituto Politécnico Nacional. México
- [15] Lichtblau, D. (2002) *Discrete optimization using Mathematica*. World multi-conference on systemics, cybernetics and informatics, International Institute of Informatics and Systemics, vol 16, pp. 169-174.
- [16] Price, K., Storn, R. y Lampinen, J. (2005) *Differential Evolution. A Practical Approach to Global Optimization*. Springer
- [17] Castro-Sánchez, N. (2010) *Un método automático para la extracción de los patrones de recepción en el español basado en diccionarios explicativos y relaciones léxicas*. Laboratorio de Lenguaje Natural y Procesamiento de Texto. Centro de Investigación en Computación. Instituto Politécnico Nacional.
- [18] Padró, L., Collado, M., Reese, S., Lloberes, M. y Castellón, I. (2010) *FreeLing 2.1: Five Years of Open-Source Language Processing Tools* Proceedings of 7th Language Resources and Evaluation Conference (LREC 2010), ELRA. La Valletta, Malta.
- [19] Kozima, H. and Furugori, T. (1993) *Similarity between words computed by spreading activation on an English dictionary*. In: *Proceedings of the 6th conference of the European chapter of ACL*, pp. 232-239.
- [20] *Longman dictionary of the English language* (1991) New edition. Edited by Brian O'Kill. Harlow: Longman.
- [21] Apresjan, J. (1974) *Regular polysemy*, *Linguistics*, 142: 5-32.

- [22] Apresjan, J. (1995) Selected works (in Russian). Moscow, V 1, 472 p., V 2, 768 p.
- [23] Diccionario Anaya de la lengua, Grupo Anaya, Internet, (1997).
- [24] Dolan, W., Vanderwende, L., and Richardson, S. (1999) "Polysemy in a Broad-Coverage Natural Language Processing System", in Polysemy: Theoretical and Computational Approaches. (ed.)
- [25] Evens, M. (ed.), (1988) Relational models of lexicon: Representing knowledge in semantic network. Cambridge: Cambridge University Press.
- [26] Mel'cuk, I. and Zholkovsky, A. (1988), "The explanatory combinatorial dictionary", in M. Evens (ed.), Relational Models of the Lexicon: Representing Knowledge in Semantic Networks. Cambridge: Cambridge University Press.
- [27] Allport, D. A. (1985) Distributed Memory, Modular Subsystems and Dysphasia, in Current Perspectives in Disphasia, Newman, S., Epstein, R.(Eds.). Edinburgh: Churchill Livingstone.
- [28] 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.
- [29] Budanitsky, A. (1999) Lexical semantics relatedness and its application in natural language processing. Technical report CSRG-390, University of Toronto, Toronto, 146 p. <ftp://ftp.cs.utoronto.ca/csrp-technical-reports/390>
- [30] Butterworth, B. (1980) Evidences from Pauses in Speech, Language Production: Speech and Talk, Vol. I. London:Academic Press, pp. 155-176.
- [31] Fellbaum, C. (1990) The English verb lexicon as a semantic net. International Journal of Lexicography 3: 278-301.
- [32] Dolan, W., Vanderwende, L., and Richardson, S. (ed.) (1999) "Polysemy in a Broad-Coverage Natural Language Processing System", in Polysemy: Theoretical and Computational Approaches.



- [33] 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.
- [34] Calvo-Castro, H. (2006) *Determinación automática de roles semánticos usando preferencias de selección sobre corpus muy grandes*. México.
- [35] Villayandre-Llamazares, M. (2010). *Aproximación a la lingüística computacional*. Departamento de filología hispánica y clásica. Universidad de León. España
- [36] Uszkoreit, H. (2002) *New Chances for Deep Linguistic Processing*, In: *Proceedings of COLING 2002*, Taipei
- [37] Gil-Londoño, N. (2006) *Algoritmos genéticos*. Universidad Nacional de Colombia Escuela de Estadística Sede Medellín.
- [38] Mezura, E. (2001) *Uso de la Técnica Multiobjetivo NPGA para el Manejo de Restricciones en Algoritmos Genéticos*. CINVESTAV-IPN. México.
- [39] Aitchison, J. (1987) *Words in the Mind. An introduction to the mental lexicon*. Oxford: Blackwell.
- [40] Fodor, J. D., Fodor, J. A., & Garrett, M. F. (1975) The psychological unreality of semantic representations. *Linguistic Inquiry* 6, pp. 515-531.
- [41] Fodor, J. A., Garrett, M.F., Walker, E.C., & Parkes, C.H. (1980) Against definitions. *Cognition* 8(3), pp. 263-367.
- [42] Goddard, C. (1991) Testing the Translatability of Semantic Primitives into an Australian Aboriginal Language. *Anthropological Linguistics* 33(1), pp. 31-56.
- [43] Goddard, C. (1995) Who are We? The natural semantics of pronouns. *Language Sciences*. 17(1), pp. 99-121.
- [44] Goddard, C. (ed.) (1997) *Studies in the syntax of universal semantic primitives*. Special Issue of *Language Sciences* 19(3).
- [45] Goddard, C. (1998) *Semantic Analysis: A Practical Introduction*. Oxford: Oxford University Press.

[46] Goddard, C. (In press) Universal units in the lexicon. In HASPELMATH, M., KÖNIG, E. OESTERREICHER, W. & RAIBLE, W. (eds.). *Language Typology and Linguistic Universals. An International Handbook*. Berlin: de Gruyter.

[47] Goddard, C. & Wierzbicka, A. (eds.) (1994) *Semantic and Lexical Universals—Theory and Empirical Findings*. Amsterdam/Philadelphia: John Benjamins.

[48] Goddard, C. & Wierzbicka, A. (eds.) (To appear) *Meaning and Universal Grammar*.