Onomázein
ISSN: 0717-1285
onomazein@uc.cl
Pontificia Universidad Católica de Chile
Chile

Pichardo Lagunas, Obdulia; Sidorov, Grigori; Cruz, Nareli; Gelbukh, Alexander
Detección automática de primitivas semánticas en diccionarios explicativos con algoritmos bioinspirados
Onomázein, núm. 29, junio, 2014, pp. 104-117
Pontificia Universidad Católica de Chile
Santiago, Chile

ONOMÁZEIN
Revista semestral de lingüística, filología y traducción

PONTIFICIA UNIVERSIDAD
CATÓLICA DE CHILE
FACULTAD DE LETRAS

Detección automática de primitivas semánticas en diccionarios explicativos con algoritmos bioinspirados

Automatic Detection of Semantics Primitives in Explicative Dictionaries with Bio-inspired Algorithms

Obdulia Pichardo Lagunas
Instituto Politécnico Nacional
México

Grigori Sidorov
Instituto Politécnico Nacional
México

Nareli Cruz Cortés
Instituto Politécnico Nacional
México

Alexander Gelbukh
Instituto Politécnico Nacional
México

ONOMÁZEIN 29 (junio de 2014): 104-117
DOI: 10.7764/onomazein.29.1

Obdulia Pichardo: Centro de Investigación en Computación, Instituto Politécnico Nacional. México.
Correo electrónico: opichardola@ipn.mx
Grigori Sidorov: Centro de Investigación en Computación, Instituto Politécnico Nacional. México.
Correo electrónico: sidorov@cic.ipn.mx
Nareli Cruz: Centro de Investigación en Computación, Instituto Politécnico Nacional. México.
Correo electrónico: nareli@cic.ipn.mx
Alexander Gelbukh: Centro de Investigación en Computación, Instituto Politécnico Nacional. México.
Correo electrónico: gelbukh@ cic.ipn.mx

Fecha de recepción: junio de 2012
Fecha de aceptación: noviembre de 2013

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 el diccionario que regresa 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 estos 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 la definición en el diccionario, y en este sentido son primitivas. En esta investigación, nuestra meta es mantener la mayor cantidad de palabras en el diccionario, es decir, tener un número mínimo de las primitivas semánticas. Presentamos un método que obtiene el conjunto de primitivas más pequeño obtenido hasta ahora. Para eso utilizamos la representación del diccionario como un grafo dirigido y aplicamos un algoritmo de evolución diferencial que determina el orden en que el grafo debe ser construido.

Palabras clave: primitivas semánticas; lexicografía computacional; computación evolutiva; evolución diferencial.

Abstract

Inevitably, any explanatory dictionary contains cycles in its definitions, that is, if a word is defined in the dictionary and then used in a definition, there is always a path in the dictionary that returns to the same word. In a good dictionary the cycles are long, but they are unavoidable. A computational dictionary cannot contain any cycles in its definitions without them affecting the ability of logical inference of computer systems. In this study, we name semantic primitives to such words in the dictionary that if removed, the cycles would be eliminated; that is, those words would not have a definition and, in this sense, they are primitive. In this research, our goal is to keep as many words in the dictionary, i.e., to minimize the number of semantic primitives. We present a method that achieves the smallest set of primitives obtained so far. In order to accomplish this, the representation of the dictionary was used as a directed graph, and a differential evolution algorithm, that determines the order in which the graph should be built, was applied to the dictionary.

Keywords: semantic primitives; computational lexicography; evolutionary computation; differential evolution.

1. Introducción

La lexicografía es la rama de la lingüística que se encarga de la compilación de diccionarios. En el caso de los diccionarios explicativos, la tarea de un lexicógrafo es describir los significados de una palabra que dependen de sus posibles usos en diferentes tipos de contextos. Tradicionalmente, el trabajo lexicográfico se hace manualmente. Sin embargo, las computadoras modernas pueden ser de gran ayuda a un lexicógrafo, y no solo en el sentido de mantener una base de datos de palabras y formatear el texto. Por ejemplo, se puede verificar automáticamente las definiciones, buscar palabras sin definición, ciclos en definiciones que sean muy cortos (y por lo tanto no deseables), proponer los candidatos a los sinónimos y antónimos, etc. (Gelbukh y Sidorov, 2003). Cabe mencionar que las computadoras hacen las sugerencias y llaman la atención a los casos problemáticos, pero siempre la decisión final la toma el lexicógrafo.

El producto tradicional de la lexicografía es un diccionario explicativo. En general, un diccionario explicativo es una colección de palabras que pertenecen a un lenguaje, y las definiciones de sus sentidos (acepciones). Nótese que se definen las palabras a través de otras palabras, y curiosamente eso lleva inevitablemente a los ciclos en las 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. En la lexicografía tradicional orientada a los humanos, deben evitarse los ciclos cortos; por ejemplo, *abeja: abeja es un insecto que segrega miel* y *miel: miel es una sustancia segregada por las abejas*. De esa manera se define *miel* a través de *abeja* y *abeja* a través de *miel*. Manualmente es relativamente fácil evitar los ciclos de longitud uno, como en el ejemplo, pero la dificultad para impedir la existencia de los ciclos crece de acuerdo con el tamaño de estos. En un buen diccionario explicativo los ciclos son largos, pero, repetimos, son inevitables (Gelbukh y Sidorov, 2010).

La lingüística de corpus también es de gran ayuda para los lexicógrafos. Esa rama de la lingüística trabaja sobre conjuntos de textos muy grandes, y determina qué tipo de información adicional hay que agregar a los corpus. Por ejemplo, se puede presentar la información confiable acerca de los usos más comunes de una palabra: los contextos de uso —lo que se llama concordancias—, las frecuencias de palabras relacionadas, etc.

La lingüística computacional (LC), que es un campo interdisciplinario entre la lingüística y la computación, específicamente la inteligencia artificial, tiene como objetivo la realización de aplicaciones informáticas que imiten la capacidad humana de hablar y entender textos escritos en lenguaje natural. Las aplicaciones desarrolladas en este campo pueden servir de apoyo a especialistas de la lengua y a usuarios comunes. Existe una rama de la lingüística computacional que se llama procesamiento de lenguaje natural (PLN), que trabaja específicamente en problemas relacionados con el procesamiento computacional, tales como traducción automática, generación automática de resúmenes, contestación de preguntas, etc. Todos estos problemas pueden ser analizados utilizando como instrumento un diccionario explicativo —que, como ya mencionamos, siempre contendrá ciclos en sus definiciones—. El problema radica en que una computadora no puede utilizar un diccionario con ciclos en sus 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.

Para poder utilizar un diccionario como herramienta computacional debemos excluir de este los posibles ciclos en las definiciones. Se busca un método que nos permita marcar algu-

nas palabras como "primitivas", es decir, en cierto sentido eliminándolas del diccionario —de la lista de los vocablos con definiciones—. Así la definición de cualquier otra palabra se expresa con las palabras primitivas, posiblemente en varios pasos. Posteriormente, se puede dar algún tratamiento especial a esas palabras primitivas, por ejemplo, sustituyéndolas por otro tipo de elementos comprensibles para los sistemas computacionales.

Sabemos que "un método natural" para la construcción de un diccionario semántico utilizable para computadoras es la definición de unas palabras a través de otras, como se hace en los diccionarios explicativos tradicionales. Esta sustitución se repite iterativamente hasta llegar a conceptos cada vez más "simples"; en algún momento es necesario detener este proceso llegando así a un conjunto de palabras "primitivas" (Rivera-Loza y otros, 2003). Esto es, se podría considerar las palabras que cierran un ciclo en las definiciones como primitivas semánticas excluyéndolas como entradas del diccionario, pero conservándolas en la construcción de las definiciones de otras entradas. De esta forma el diccionario es mantenido sin ciclos de cualquier longitud.

En este trabajo, el diccionario explicativo será representado como un grafo dirigido para encontrar las palabras primitivas. Las palabras, ya sean entradas del diccionario o definidoras, se representan como vértices. Las aristas indicarán el tipo de relación que existe entre los vértices. La idea básica de esta solución consiste en construir el grafo (diccionario) empezando desde cero agregando nodo tras nodo —utilizando un algoritmo específico de construcción—. El grafo se mantendrá sin ciclos siempre (durante toda su construcción). La parte más importante del algoritmo es cómo definir en qué orden se introducirán las palabras. Con este fin se utiliza una técnica de inteligencia artificial —el orden de inserción de las palabras (nodos) en el grafo es definido mediante el uso de un algoritmo de evolución diferencial—. Este algoritmo generará diferentes permutaciones de entrada. Su función objetivo es minimizar el número de palabras consideradas primitivas semánticas.

2. Trabajos relacionados

Propuesto por Anna Wierzbicka, el metalenguaje semántico natural (*NSM* por sus siglas en inglés) es un conjunto de palabras con las cuales se puede definir el resto de las palabras de una lengua (Wierzbicka, 1980). La teoría de Wierzbicka propone un número de alrededor de 60 palabras consideradas como primitivas. De acuerdo a esta teoría el significado de cualquier expresión se puede especificar a través de una paráfrasis reductiva. En lenguaje ordinario, todo idioma debe tener un núcleo semántico irreductible, además de reglas para su combinación. El núcleo debe ser el mismo para todas las lenguas (Wierzbicka, 1996).

Apresjan (1974: 5-32; 1995) apoya la idea de usar un vocabulario restringido considerado la base para construir las definiciones del resto de las voces de una lengua, pero afirma que este no puede ser tan pequeño como el marcado por Wierzbicka.

El *Longman Dictionary of Contemporary English, LDOCE* (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 restringido. El tamaño de este vocabulario es de alrededor de 2.866 palabras, a las cuales corresponden 2.206 lemas. En el caso del *LDOCE* el vocabulario definidor fue construido utilizando como base la Lista de Servicios Generales de Michael West (*Michael West's General Service List*), que fue creada en 1953 y está compuesta por las palabras con mayor frecuencia en un corpus de inglés escrito.

Kozima y Furugori crearon una red semántica para el *LDOCE* en la que cada palabra del diccionario es representada por un nodo. Creando

un sistema cerrado en el que todas las palabras que se usen en una definición serán a su vez definidas por el mismo diccionario, concluyeron que "si existe un vocabulario definidor, le corresponde la parte más densa de la red, mientras que las palabras que no son definidoras no están ligadas entre sí, por lo tanto se encuentran en la periferia" (1993: 232-239).

Rivera-Loza y otros (2003) retoman este problema. Representan al diccionario como un grafo dirigido que 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. 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. La presente investigación retoma el trabajo implementando el uso de algoritmos evolutivos con el objetivo de minimizar el tamaño del conjunto obtenido y maximizar la calidad de las palabras seleccionadas.

3. Definiciones

A continuación se describen algunas definiciones que consideramos necesarias para el planteamiento del problema que nos ocupa. Las usamos para tener una representación formal del problema, si el lector no se siente suficientemente seguro en ese campo, también presentamos las explicaciones intuitivas del problema, lo que permite saltar las definiciones formales.

3.1. Conceptos de teoría de grafos

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

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

$V_1, V_2, \ldots V_n$

Diremos que una trayectoria dirigida es cerrada si y solo 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 $G = (V, F)$ un grafo dirigido, entonces $G_1 = (V_1, F_1)$ es un subgrafo de G si

$V_1 \neq \emptyset$ y $F_1 \subset F$,

donde toda arista de F_1 es incidente con los vértices de V_1.

Los grafos pueden representarse usando una lista de adyacencia que consiste en 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 1.

En el grafo de la ilustración 1 pueden observase diferentes ciclos; el más largo de ellos es el conformado por la siguiente trayectoria cerrada: A, B, E, D, C, A.

Para nuestro problema, los vértices (A, B, C, D, E) representarán las palabras del diccionario y las aristas indicarán si la palabra está contenida en la definición.

3.2. Algoritmo de evolución diferencial

El algoritmo de evolución diferencial (ED) es un método estocástico de optimización perteneciente a la categoría de computación evolutiva. Este algoritmo fue creado con el fin de simular procesos de evolución natural. El diagrama ge-

ILUSTRACIÓN 1

Lista de adyacencia

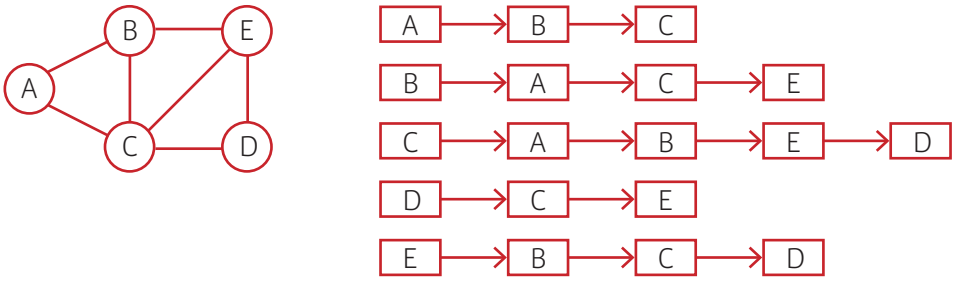


ILUSTRACIÓN 2

Descripción general del algoritmo de evolución diferencial

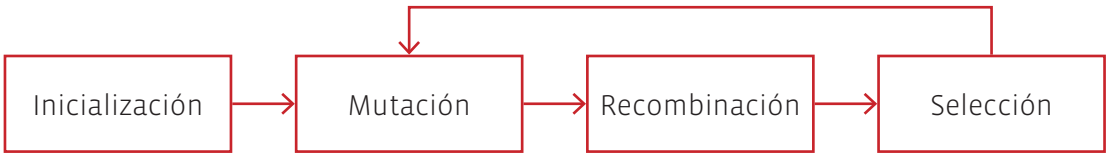


ILUSTRACIÓN 3

Algoritmo de evolución diferencial

1 Inicio
2 G=0
3 Genera población inicial $\vec{x}_{i,G} \forall\, i, i = 1,\ldots,NP$
4 Evalúa $f(\vec{x}_{i,G})\ \forall i, i = 1,\ldots,NP$
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 ⇒ $u_{i,j,G+1} = x_{r3,j,G} + F(x_{r1,j,G} - x_{xr2,j,G})$
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

neral del algoritmo puede verse en la ilustración 2. Al igual que otros algoritmos de esta categoría, la ED posee una población de individuos (en este caso arreglos de permutaciones) que representan las posibles soluciones al problema. Estos se mezclan y mutan para producir nuevos individuos, los cuales serán elegidos de acuerdo a su desempeño (Fogel y otros, 1966). 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" (Mezura-Montes y otros, 2006). El algoritmo se muestra en la ilustración 3.

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.

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 (Lichtblau, 2002).

3.2.1. Permutaciones con evolución diferencial

En esta sección explicamos un detalle técnico del algoritmo de ED. 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 se representan por palabras del diccionario asociadas a un número entero. Así el algoritmo de ED representado en la ilustración 3 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 (Price y otros, 2005).

Veamos un ejemplo:

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

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

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

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

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:

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

4. Método propuesto de detección de primitivas semánticas

Se busca una permutación de la lista de palabras que conforman el diccionario que minimi-

ce el conjunto de palabras primitivas. Ya que el diccionario ha sido representado como un grafo dirigido que será construido nodo a nodo, buscamos el orden en el que los nodos deben ser insertados.

Sabemos que el número de posibles permutaciones de una lista corresponde al factorial del total de los elementos que la componen, es decir, *n!* La cantidad de palabras que conforma el diccionario es de 30.971, por lo que el total de posibles permutaciones corresponde a 30.971! Considerando lo anterior podemos darnos cuenta de que contabilizar el total de permutaciones de entrada de palabras al grafo es incomputable. En ciencias de la computación un problema como el antes mencionado que no se puede resolver con un cálculo directo se denomina de tipo *NP*-completo.

Los métodos heurísticos como los algoritmos de computación evolutiva son herramientas indispensables en la solución de problemas de tipo *NP*-completo. Por lo anterior proponemos el uso de un algoritmo de evolución diferencial.

El algoritmo ED proporcionará diferentes permutaciones con las que se construirá el grafo correspondiente al diccionario palabra por palabra verificando después de cada inserción la posibilidad de aparición de un nuevo ciclo como sigue:

> Dado σ, que es una permutación de entrada generada por un algoritmo de evolución diferencial, el grafo G_1 se construye palabra por palabra. Inicialmente, G_1 está vacío. Se construye insertando uno a uno los vértices y sus relaciones hacia los vértices ya insertados. Con cada inserción en G_1 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. 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. La heurística utilizada es un algoritmo de evolución diferencial.

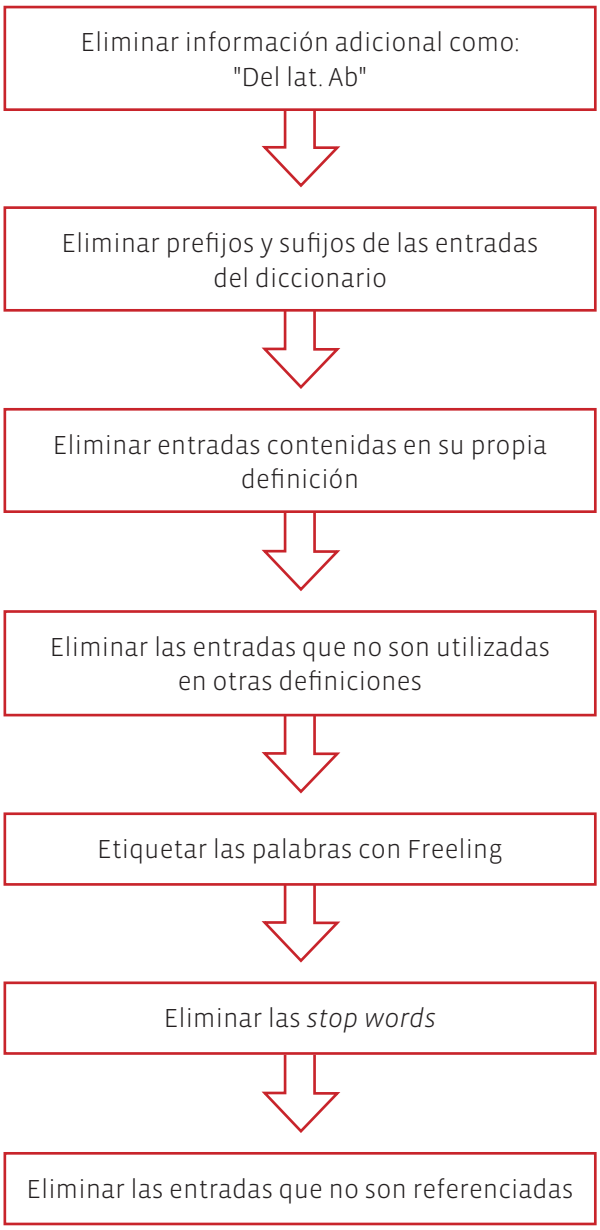
4.1. Preprocesamiento del diccionario

Se utilizó el diccionario de la lengua española del grupo Anaya, edición 1996. Este diccionario contiene 30.971 entradas. Se eliminaron los sufijos y afijos de la lista de las definiciones, lo que redujo la lista de palabras a 30.725. El análisis del diccionario se realizó por cada definición en el diccionario, así que las diferentes acepciones de cada entrada fueron agrupadas en una bolsa de palabras.

Las palabras del diccionario se lematizaron y etiquetaron utilizando Freeling (Padró y otros, 2010), que es una herramienta de análisis morfológico y sintáctico automático libremente disponible desarrollada por la Universidad Politécnica de Cataluña. Los experimentos utilizan únicamente las palabras significativas —que aportan información semántica a la definición, tales como verbos, adverbios, sustantivos y adjetivos—. Se eliminaron las palabras que no coincidieran con estas categorías, es decir, las palabras auxiliares (*stop words*, en inglés), como preposiciones, artículos, etc. También se eliminaron aquellas entradas que estaban contenidas en su propia definición, es decir, aquellas que formaban lazos en el grafo —un lazo es formado por un nodo del grafo que tiene una arista que apunta hacia el mismo—, considerándolas como primitivas.

Se extrajeron de la lista aquellas palabras que no eran utilizadas en el resto de las definiciones porque no tendrían la posibilidad de cerrar algún ciclo y por lo tanto no tienen oportunidad de considerarse como primitivas semánticas.

ILUSTRACIÓN 4
Preprocesamiento del diccionario



Las tareas anteriores fueron realizadas con el fin reducir en lo posible el número de entradas al grafo y así evitar que este crezca excesivamente. 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. El diagrama de preprocesamiento puede visualizarse en la ilustración 4.

4.2. Aplicación de evolución diferencial

El algoritmo de ED descrito arriba requiere 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:

$$X_i^m = [1\ 2\ 3\ 4\ 5 \ldots n]$$

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

El algoritmo de ED tendrá 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 estas, 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 se definió de la siguiente forma:

$$\text{Minimizar } |P|, \text{ donde } P\{x \mid x \in V \wedge x \notin G1\}$$

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

5. Diseño de los experimentos

La lista de las palabras que se usan para la detección de las primitivas semánticas se representó como una lista de adyacencia y tiene un total de 9.820 palabras, cada una de estas con sus relaciones —palabras que participan en su definición—. El algoritmo generó diferentes permutaciones de esta lista que sirvieron como población inicial y que fueron recombinadas durante cada iteración de este. Se generó el grafo para cada una de estas recombinaciones verifi-

cando la ausencia de ciclos.

En algunas ejecuciones la ED fue configurada para utilizar elitismo, 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.

El algoritmo fue ejecutado en 33 ocasiones ajustando en cada una la configuración de los parámetros. Ejemplos de los parámetros se presentan en la tabla 1.

Para el algoritmo propuesto, 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 el conjunto *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 de experimentos. El primero consistió en reducir el tamaño del conjunto de primitivas obtenido mediante el uso del algoritmo de evolución diferencial, es decir, se obtienen los datos del conjunto (la lista) de las primitivas semánticas.

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

El segundo experimento está relacionado con la evaluación del conjunto obtenido. Consiste en medir la intersección existente entre el conjunto de palabras obtenidas por el algoritmo y una traducción al español del vocabulario de *LDOCE*. La traducción contempló todas las posibles acepciones de un vocablo, agrupándolas como una bolsa de palabras.

Ya que el vocabulario de *LDOCE* es un conjunto creado manualmente por lexicógrafos que tomó años de investigación, creemos que entre más grande sea el número de coincidencias mayor será la calidad de la lista.

El tercer experimento también evalúa la calidad del conjunto obtenido. El 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 vocabu-

TABLA 1
Configuración de parámetros

	Número de individuos	Número de generaciones	Elitismo	Porcentaje de mutación	Número de primitivas
Ejecución 1	300	500	No	0.15	2.263
...	...	...	...	...	...
Ejecución 23	500	500	No	0.1	2.259
...	...	...	...	...	...
Ejecución 32	300	500	Sí	0.2	2.179
Ejecución 33	500	300	Sí	0.2	2.169

lario 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 (1993), 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.

6. Análisis de resultados

El algoritmo fue ejecutado en 33 ocasiones con diferentes configuraciones en los parámetros. Dada la naturaleza estocástica de los algoritmos evolutivos, se calcula que después de 30 ejecuciones (corridas) los resultados arrojados ya cuentan con validez estadística. Esto es, los buenos resultados de una sola corrida no significan que el algoritmo funciona, ya que pudo ser producto de la casualidad, pero si después de 30 corridas los resultados que se obtienen siguen siendo apropiados se considera que ya no es casual.

La configuración que obtuvo mejores resultados fue:

- 500 individuos,
- 300 generaciones,
- CR=0.2 (el grado de recombinación entre individuos de la población),
- F=0.5 (el valor de mutación de los vectores de prueba).

En la tabla 2 presentamos la comparación entre el mejor resultado obtenido y el promedio de resultados de las corridas. Recordamos que el número de palabras en el vocabulario de *LDOCE* es de 2.851. El promedio de los resultados obtenidos arrojó un total de 2.581 palabras, que tiene 412 más que el mejor resultado obtenido y 270 palabras menos que el vocabulario *LDOCE*. También se puede observar la evaluación del conjunto obtenido y el conjunto promedio con respecto al vocabulario *LDOCE*. El porcentaje de las coincidencias absolutas se calcula dividiendo el número de primitivas en *LDOCE* entre el tamaño del *LDOCE*; esta medida da idea de qué porcentaje de *LDOCE* está cubierto por el conjunto obtenido. El porcentaje de las coincidencias relativas se calcula dividiendo el número de primitivas en *LDOCE* entre el tamaño del conjunto de primitivas. Este porcentaje muestra qué parte de las primitivas están en *LDOCE*.

Del mejor conjunto obtenido, poco más de la mitad de las palabras (55,9%) coincide con vocabulario *LDOCE*, es decir, un método completamente automatizado logró identificar más del 50% de vocabulario sugerido por especialistas humanos.

Analizando el caso promedio, podemos ver que el número de coincidencias crece, lo que indica que los métodos automatizados pueden encontrar cierto grupo de palabras que pertenecen al vocabulario pero cuya permanencia en el grafo depende de la presencia de otras palabras.

TABLA 2
Resultados obtenidos: el mejor resultado vs. el promedio

	Número de primitivas	Coincidencias absolutas con *LDOCE*	Coincidencias relativas con *LDOCE*
Mejor resultado obtenido	2.169	1.594 (55,9% del conjunto)	73,5%
Promedio de los resultados obtenidos	2.581	1.652 (57,9% del conjunto[1])	64,0%

1 Este porcentaje crece porque el número de palabras del conjunto es mayor; se observa que la diferencia promedio es menor en la columna de las coincidencias relativas.

Dado lo anterior, podemos pensar que existe un conjunto de palabras más representativas que mantienen una fuerte relación con palabras de menor "fuerza", pero necesarias. La búsqueda de estas palabras y sus conexiones puede ser una alternativa para complementar el conjunto de primitivas.

En la tabla 3 presentamos una comparación del conjunto obtenido por el algoritmo de evolución diferencial con el método aleatorio —conjunto generado por el algoritmo aleatorio (Rivera-Loza y otros, 2003). Nuestro conjunto contiene 2.169 palabras (incluido el conjunto de palabras que generaban lazos), 77 palabras menos que el conjunto de comparación.

TABLA 3

Resultados obtenidos: comparación con el otro conjunto

	Número de primitivas	Coincidencias absolutas con *LDOCE*	Coincidencias relativas con *LDOCE*
Algoritmo de ED	2.169	1.594 (55,9%)	73,5%
Alg. aleatorio	2.246	1.487 (52,15%)	66,2%

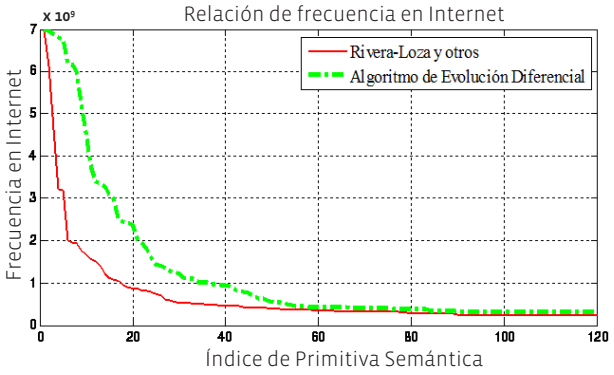
Los resultados obtenidos por el algoritmo de evolución diferencial mejoran en un 7,3% el tamaño del conjunto aleatorio, lo que es la diferencia entre las coincidencias relativas en la tabla 3.

Si bien la diferencia absoluta entre los conjuntos es pequeña, se puede observar como el uso de heurísticas en la solución de problemas de PLN proporciona mejores resultados. El algoritmo de ED fue estructurado para evaluar una sola variable —el tamaño del conjunto—; si se consideran otras posibles variables como la frecuencia de las palabras o la conexión entre palabras, es posible que el algoritmo proporcione mejores resultados.

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 5, el conjunto alcanzado con el algoritmo de evolución diferencial se mantiene siempre por encima del conjunto aleatorio. El eje X representa el rango de la primitiva semántica, el eje Y representa su frecuencia (en este caso multiplicado por 109 para aparecer en la gráfica). 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%.

ILUSTRACIÓN 5

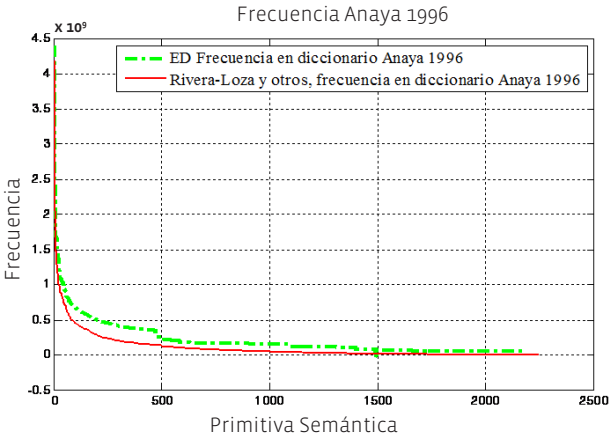
Frecuencia absoluta en Internet



Otro experimento calcula las repeticiones de las palabras en el propio diccionario. La frecuencia obtenida para el diccionario Anaya muestra un comportamiento heterogéneo; con el procedimiento 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 6.

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.

ILUSTRACIÓN 6
Frecuencias en el mismo diccionario



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.

7. Conclusiones

En este artículo presentamos un método para construir un conjunto de palabras que al considerarse primitivas semánticas permiten obtener un diccionario explicativo sin ciclos en definiciones.

La implementación de un algoritmo de evolución diferencial permitió mejorar los resultados obtenidos por trabajos anteriores en un 7,3% respecto al tamaño del conjunto obtenido.

El algoritmo de evolución diferencial no solo minimizó el conjunto de primitivas, también logró obtener un conjunto de palabras con mayor calidad, según los criterios establecidos. Se obtuvo un conjunto de palabras cuya frecuencia en el uso habitual de la lengua es mayor que la que se obtuvo con otro método.

Los resultados obtenidos muestran que es posible la existencia de un conjunto de palabras cuyas conexiones proporcionen las palabras restantes del conjunto de primitivas.

Se pudo observar que el comportamiento de un diccionario representado como un grafo dirigido es muy similar al de otros problemas de computación, lo que proporciona diferentes metodologías para la obtención de posibles soluciones a los problemas de diccionarios.

Se cuenta con un diccionario sin ciclos, que al ser complementado puede ser una herramienta eficaz en el tratamiento de problemas de PLN.

Se sabe que, dadas las características de los problemas que trata de resolver, las heurísticas no proporcionan el mejor resultado. Pero sí permiten obtener el mejor resultado encontrado de entre muchas posibilidades, es decir, evaluando muchos posibles conjuntos de respuestas.

Se pudo observar que un método completamente automatizado puede obtener resultados aceptables (aunque no perfectos) en tareas consideradas exclusivas para humanos. De esa manera, los resultados de un sistema automatizado pueden ser útiles como un punto de partida para los lexicógrafos humanos.

El análisis de diccionarios es un campo poco explorado hasta el momento, pero ya puede proporcionar herramientas que apoyen las técnicas de procesamiento de lenguaje natural y la creación lexicográfica, de ninguna manera sustituyendo a los lexicógrafos, sino ayudándolos.

Existen diversas limitaciones para este experimento; por ejemplo, el vocabulario con el que el conjunto obtenido fue comparado tuvo que ser traducido del idioma inglés, lo que implica la posible omisión de algunas palabras o de sentidos específicos. El algoritmo de evolución diferencial no puede asegurar proporcionar el mejor resultado aunque haya mejorado los ya obtenidos. El vocabulario expresado en los diccionarios suele abreviarse o cambiarse por el uso de regionalismos que pueden afectar los resultados de la aplicación.

8. Trabajo futuro

Algunas de las direcciones para el trabajo futuro son:

- Efectuar cambios en la función objetivo del algoritmo de evolución diferencial para incluir otros criterios de evaluación para las poblaciones del algoritmo.
- Experimentar con diferentes configuraciones de parámetros.
- Realizar los experimentos descritos en este documento con varios diccionarios más; esto nos permitirá comparar las listas obtenidas por el mismo algoritmo pero con diferentes diccionarios de entrada.
- Comparar los diferentes conjuntos obtenidos con vocabularios creados para otras lenguas, así validando el concepto de universalidad de los conjuntos de primitivas.
- Realizar un análisis lexicográfico de los vocabularios con el fin de verificar su aplicabilidad en herramientas lingüísticas.

9. Bibliografía citada

Apresjan, Juri, 1974: "Regular polysemy", *Linguistics* 142, 5-32.
Apresjan, Juri, 1995: *Selected works (in Russian)*, 2 vols., Moscow.

Fogel, Lawrence Jerome, Alvin J. Owens y Michael John Walsh, 1966: *Artificial Intelligence through Simulated Evolution*, New York: Wiley & Sons, Inc.

Gelbukh, Alexander y Grigori Sidorov, 2003: "Hacia la verificación de diccionarios explicativos asistidos por computadora", *Estudios de Lingüística Aplicada* 38, 89-108.
Gelbukh, Alexander y Grigori Sidorov, 2010: *Procesamiento automático del español con enfoque en recursos léxicos grandes*, México: IPN.

Grupo Anaya, 1996: *Diccionario Anaya de la lengua*.

Kozima, Hideki y Teiji Furugori, 1993: "Similarity between words computed by spreading activation on an English dictionary" en *Proceedings of the 6th conference of the European chapter of ACL*, 232-239.

Lichtblau, Daniel, 2002: "Discrete optimization using Mathematica" en *World multi-conference on systemics, cybernetics and informatics*, v. 16, Orlando, Florida: International Institute of Informatics and Systemics, 169-174.

Mezura-Montes, Efrén y otros, 2006: "A Comparative Study of Differential Evolution Variants for Global Optimization" en *Genetic and Evolutionary Computation Conference (GECCO)*.

Padró, Lluís y otros, 2010: "FreeLing 2.1: Five Years of Open-Source Language Processing Tool" en *Proceedings of the 7th Language Resources and Evaluation Conference*, La Valletta, Malta.

Pearson Education, 1991: *Longman Dictionary of Contemporary English*, London.

Price, Kenneth, Rainer Storn y Jouni Lampinen, 2005: *Differential Evolution. A Practical Approach to Global Optimization*, Springer.

Rivera-Loza, Gabriela, Alexander Gelbukh y Grigori Sidorov, 2003: *Selección automática de primitivas semánticas para un diccionario explicativo del idioma español*. Tesis de maestría, CIC-IPN, México.

Wierzbicka, Anna, 1980: *Lingua Mentalis: The semantics of natural language*, New York: Academic Press.
Wierzbicka, Anna, 1996: *Semantics: Primes and Universals*, Oxford: Oxford University Press.