Descrubrimiento incremental de patrones secuenciales para la inferencia de gramáticas

Ramiro Aguilar
Instituto de Investigaciones en Informática
Universidad Mayor de San Andrés
Av. Villazón 1995, La Paz, Bolivia
ramiro@tejo.usal.es

Luis Alonso, Vivian López, María N. Moreno
Departamento de Informática y Automática
Universidad de Salamanca
Plaza de la Merced S/N, 37008 Salamanca
{lalonso, vivian, mmg}@usal.es

Resumen

En este trabajo se describe una metodología para generar una gramática a partir de datos textuales. Se presenta una técnica de descubrimiento incremental de patrones secuenciales que obtiene reglas de producción que después de ser simplificadas y compactadas con criterios bioinformáticos conforman una gramática que no solo reconoce el conjunto de datos inicial sino también los datos extendidos.

1. Introducción

La creciente cantidad de información en forma de documentos hace que el análisis sobre los mismos sea cada vez más complejo, con lo cual se necesitan métodos automáticos e inteligentes para su tratamiento y comprensión. Para comprender qué dicen los datos, se necesita conocer la estructura del lenguaje que ellos tienen. Para contribuir a este fin, en [14] se define un esquema general que propone un método de minería de datos sobre texto, para descubrir el conocimiento sintáctico-semántico del mismo. Como parte de todo el proceso que propone, se plantea realizar la obtención de la gramática base a partir de los patrones secuenciales obtenidos en el texto, es decir la inferencia gramatical del lenguaje del texto.

En el presente trabajo se describe un proceso nuevo de minería de datos que combina técnicas híbridas del análisis de asociación y el enfoque de los algoritmos de secuenciación clásicos de genómica para generar estructuras gramaticales a partir de un lenguaje determinado. Posteriormente, estas estructuras son convertidas a *gramáticas independientes de contexto*. Inicialmente el método se aplica lenguajes independientes de contexto con la posibilidad de aplicar, posteriormente, a los lenguajes de programación estructurados, el lenguaje del libro de la vida plasmado en genomas y proteomas y, conciertas limitaciones, los lenguajes naturales.

1.1. El problema de la inferencia gramatical

La inferencia gramatical (IG) es un área transversal que involucra los campos del aprendizaje automático, la teoría de lenguajes formales, el reconocimiento de patrones sintácticos y estructurales, la biología computacional, el reconocimiento del habla y otros [6][5].

El problema de la IG es el aprendizaje de una descripción del lenguaje a partir de datos. El problema de la inferencia de lenguajes independientes de contexto involucra cuestiones teóricas y prácticas. Los aspectos prácticos incluyen el reconocimiento de patrones y el reconocimiento del habla; uno de los enfoques del reconocimiento de patrones es la inferencia de gramáticas independientes de contexto (GIC) que produzcan un conjunto de patrones [9]; otro enfoque busca la habilidad para inferir GIC a partir de lenguajes naturales lo que permitiría a un reconocedor del habla mo-

Actas del III Taller Nacional de Minería de Datos y Aprendizaje, TAMIDA2005, pp.85-95
ISBN: 84-9732-449-8

dificar su gramática interna sobre la marcha, permitiéndole ajustarse a los hablantes independientes. Los aspectos teóricos tienen importancia en lo concerniente a las limitaciones serias de los lenguajes independientes de contexto [13] y, en definitiva, para construir algoritmos de aprendizaje factibles que imiten el modelo del lenguaje humano (este último motivo puede ser cuestionable pero es una de las primeras motivaciones de la inferencia gramatical [11]).

El aprendizaje de un lenguaje también tiene que ver con su identificación. En la literatura de la inferencia gramatical, la atención se enfoca sobre la *identificación en el límite*, de esta manera, en cada instante de tiempo t la máquina que aprende recibe una unidad de información i_t sobre un lenguaje y devuelve una hipótesis $H(i_1, ..., i_t)$; el algoritmo de aprendizaje funciona si después de una cantidad finita de tiempo, todas sus conjeturas son las mismas para una descripción correcta en el lenguaje de la pregunta. Otro criterio de aprendizaje es la *identificación exacta usando consultas en tiempo polinomial*, en este contexto, la máquina de aprendizaje tiene acceso a los oráculos que pueden responder preguntas y lo hacen en un tiempo polinomial con una descripción correcta del lenguaje [13].

Hace casi 20 años atrás que una característica problemática sigue presente dentro de la inferencia gramatical, aquella que plantea que las aplicaciones de inducción gramatical no están resueltas [17][16]. El trabajo detallado en [6][7] aunque plantea que en la práctica los algoritmos con propiedades matemáticas obtienen mejores resultados que los algoritmos con propiedades heurísticas, es decir cuando se usan autómatas finitos o, por su parte, cuando se construyen algoritmos de aprendizaje de GIC, también puntualiza que para el enfoque heurístico no existe un "benchmark" común y es entonces más dificil comparar y evaluar la efectividad de estos métodos. Con lo mencionado anteriormente, nuestra propuesta dispone de un conjunto de datos disponible con el cual es validada y estamos abiertos a otras comparaciones para mejorar/ratificar nuestro trabajo.

2. Técnicas para el análisis de asociación

El análisis de asociación involucra técnicas que difieren en su forma de operación pero que buscan encontrar relaciones entre los atributos de un conjunto de datos. Algunas técnicas son:

- Reglas de asociación
- Descubrimiento de patrones secuenciales, y
- Descubrimiento de asociaciones

2.1. Reglas de asociación

Las reglas de asociación (RA) describen las relaciones de ciertos atributos respecto de otros en una base de datos (BDs). Estas reglas identifican implicaciones de causa y efecto entre los diferentes atributos de la BDs. Por ejemplo, en el registro de compras de productos, se identifica qué artículo de compra está relacionado con otro; por ejemplo: "el 80 % de las personas que compra pañales para bebé, también compra talco".

Una regla tiene la forma "si X entonces Y" o bien $X \Rightarrow Y$. X se llama *antecedente* de la regla (en el ejemplo, "compra pañales"); Y se llama *consecuente* de la regla (en el ejemplo, "compra talco").

La generación de la regla se fundamenta por aspectos estadísticos y probabilísticos como el factor de soporte (f_s), el factor de confianza (f_c) y el factor de confianza esperada (f_e) definidos como: $f_s = \frac{nro\_veces\_regla}{nro\_total\_registros}$, $f_c = \frac{nro\_veces\_regla}{nro\_veces\_X}$ y $f_e = \frac{nro\_veces\_Y}{nro\_total\_registros}$.

El valor mínimo del factor de soporte para las reglas debe ser mayor que un umbral predeterminado. Si el factor de confianza de una regla es mayor a 0.5, entonces la regla aparece, por lo menos, en la mitad del número de instancias a las que afecta con lo que la regla tiene sentido considerable. La diferencia entre el factor de soporte y el factor de confianza esperada debería ser mínima para asegurar la efectividad de la regla.

Por ejemplo, consideremos los datos del cuadro 1, una regla obtenida es $A = 2 \Rightarrow B =$

2 con $f_s = 0{,}43$, $f_c = 0{,}75$ y $f_e = 0{,}57$, que dice que el 75 % de las instancias con $A = 2$ implican $B = 2$, además en el 43 % de todas las instancias se cumple dicha regla y $B = 2$ aparece en el 57 % de todas las instancias.

A	B	C	D	E	F
2	2	6	0	1	0.2
2	2	5	0	1	0.2
2	2	6	1	1	0.2
3	2	7	1	0	0.8
2	3	8	1	0	0.8
3	3	8	1	0	0.8
3	3	7	1	0	0.8

Cuadro 1: Conjunto de datos.

2.2. Descubrimiento de asociaciones

De forma similar a las RA, el descubrimiento de asociaciones (DA) trata de encontrar implicaciones entre diferentes pares atributo-valor de modo que la aparición de estos determine una asociación presente en una buena cantidad de los registros de la BDs. Para descubrir asociaciones se realizan los siguientes pasos:

1. Asociar un identificador a cada transacción o registro
2. Ordenar secuencialmente las transacciones según su identificador
3. Cuantificar las ocurrencias de los artículos creando un vector en el que se contabiliza cada artículo. Aquellos elementos en los que la cuenta está por debajo de un "umbral", se eliminan
4. Combinar en una matriz las transacciones atributo-valor y realizar el conteo de ocurrencias eliminando aquellos elementos que no superen el umbral
5. Repetir sucesivamente los pasos 3 y 4 hasta no poder combinar más las transacciones

Con los datos del cuadro 1, estableciendo el *umbral* = 2, se aplica la técnica como se observa en la figura 1 y se generan las siguientes asociaciones:

1. $A2 \Rightarrow B2 \Rightarrow E1 \Rightarrow F0{,}2$
2. $A3 \Rightarrow D1 \Rightarrow E0 \Rightarrow F0{,}8$
3. $B3 \Rightarrow D1 \Rightarrow E0 \Rightarrow F0{,}8$

Las asociación 1 indica: si el valor para A y B es 2 y el valor para E es 1 y el valor para F es 0.2, entonces los registros con esas características pueden pertenecer a una clase. Las demás asociaciones muestran las posibles características de los registros para pertenecer a otra clase o comportamiento.

2.3. Descubrimiento de patrones secuenciales

El descubrimiento de patrones secuenciales (DPS) es muy similar a las RA pero busca detectar patrones entre transacciones de forma que la presencia de un conjunto de items preceda a otro conjunto de items en una BDs durante un periodo de tiempo. Por ejemplo, si los datos corresponden a registros de compras de artículos por parte de clientes, se puede obtener una descripción de qué artículos compra frecuentemente un cliente, y sobre todo, cual es la secuencia de su compra. Así, la próxima vez, se conocería el perfil del cliente, y se podrá predecir la secuencia de su compra. Este criterio se puede aplicar a otro dominio de datos, por ejemplo, en el contexto de la bioinformática, cuando los datos a tratar corresponden a la cadena de bases nucleótidas del genoma y se descubren secuencias como patrones que codifican genes que conforman alguna proteína [1] [8].

El DPS tiene el siguiente funcionamiento:

1. Identificar el atributo relacionado con el tiempo
2. Considerando el periodo de tiempo para el que se quiere descubrir los patrones secuenciales, crear una tabla ordenada por el identificador de la transacción
3. Crear otra tabla uniendo los artículos de compra de cada cliente

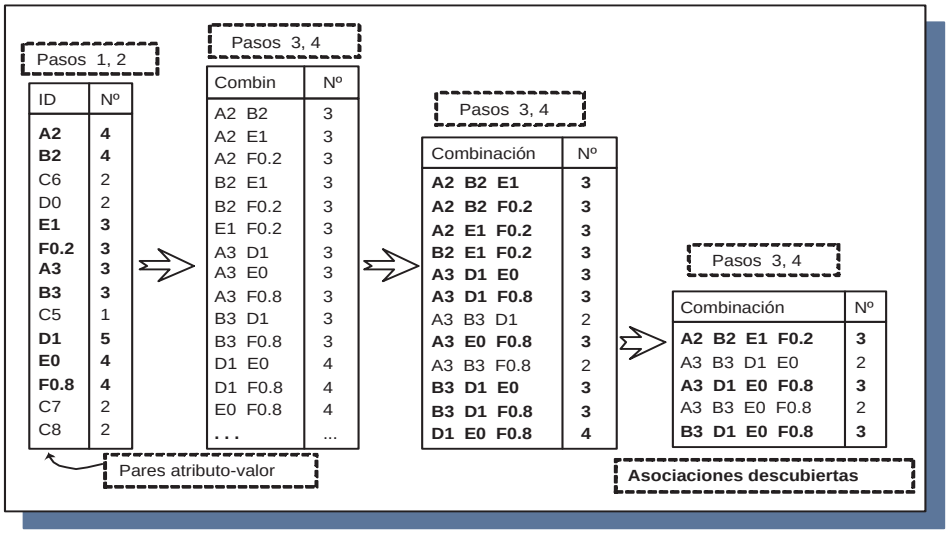

Figura 1: Descubrimiento de asociaciones en los datos del cuadro 1 (con *umbral* $\geq$3).

4. Inferir los patrones secuenciales cuyo factor de soporte supere un valor umbral.

Los patrones descubiertos muestran instancias de artículos que aparecen en forma consecutiva en el conjunto total de datos como se aprecia en el ejemplo de la figura 2.

3. Gramáticas, lenguajes y bioinformática

3.1. Gramática independiente de contexto

Una gramática $\mathcal{G}$ está definida como $\mathcal{G} = (\mathcal{N}, \mathcal{T}, \mathcal{P}, \mathcal{S})$, donde $\mathcal{N}$ es el conjunto de símbolos no terminales, $\mathcal{T}$ es el conjunto de símbolos terminales o categorías sintácticas, $\mathcal{P}$ es el conjunto de producciones y $\mathcal{S}$ es el símbolo inicial [15]. El lenguaje de una gramática llamado $\mathcal{L}(\mathcal{G})$ es el conjunto de todas las cadenas terminales w que tienen derivaciones desde el símbolo inicial. Es decir: $\mathcal{L}(\mathcal{G}) = \{w \; esta \; en \; \mathcal{T}^* \mid \mathcal{S} \Rightarrow^* w\}$

Una gramática independiente de contexto (GIC) tiene reglas de producción del tipo $A \rightarrow \alpha$ donde $A \in \mathcal{N}$ y $\alpha \in (\mathcal{N} \cup \mathcal{T})^*$. La sustitución de A por α se realiza independientemente del lugar en el que aparezca A [15]. La mayor parte de los lenguajes de programación están generados por gramáticas de este tipo (aumentadas con algunos elementos contextuales necesarios para la semántica del lenguaje).

3.2. Gramáticas y bioinformática

La Bioinformática emplea tecnologías informáticas y computacionales para desarrollar métodos, estrategias y programas que permitan manejar, ordenar y estudiar la cantidad inmensa de datos biológicos que se han generado y se generan en la actualidad. Por ejemplo, para el genoma humano (GH), la bioinformática busca encontrar sentido al lenguaje de los más de 37.000 millones de pares A, C, T y G que se han compilado y almacenado en el "libro de la vida".

Merecen la ocasión de ser entendidas las BDs gigantescas que contienen los detalles de las circunstancias de tiempo y lugar en que se activan los genes, la conformación de las proteínas que especifican, la forma en que influyen unas proteínas sobre otras y el papel que tales influjos puedan desarrollar en las enfermedades. Además, ¿cuáles son las rela-

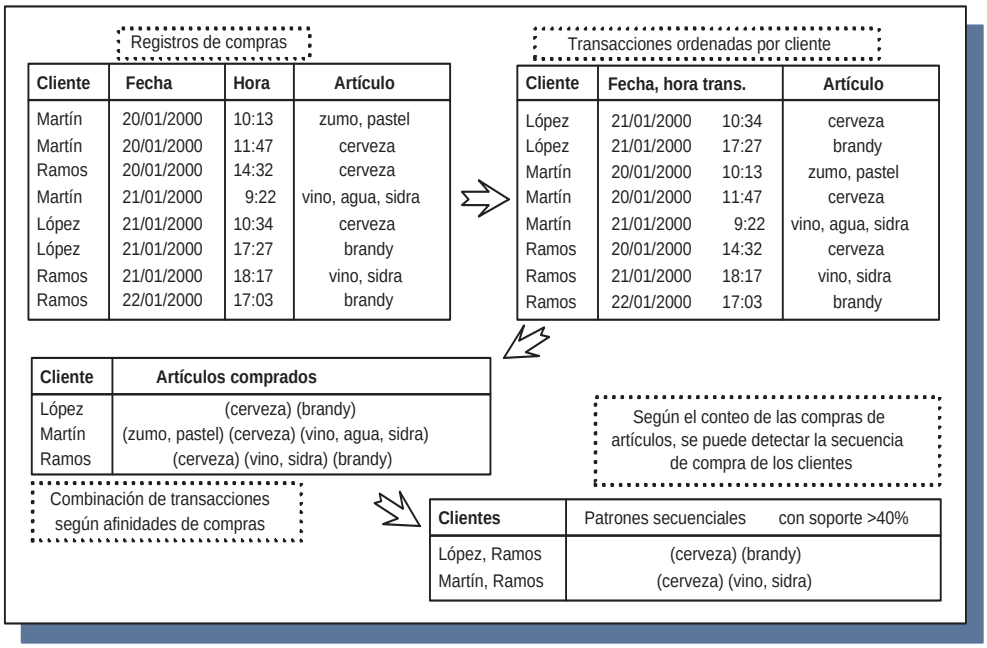

Figura 2: Descubrimiento de patrones secuenciales en registros de compras (elaborado en base a [4]).

ciones del GH con los genomas de los organismos modelo, como la mosca de la fruta, los ratones y las bacterias? ¿Se podrá descubrir patrones secuenciales que muestren cómo se relacionan entre sí los fragmentos de información? y ¿se podrá conformar una estructura gramatical que muestre la interpretación del conjunto resultante? Si logramos inferir esa estructura para este tipo de lenguaje coadyuvaremos a comprender la función real de la estructura del ADN y entenderemos algo más de los cuestionamientos planteados.

Una de las aplicaciones de la bioinformática es la farmacología, brindando soluciones renovadoras al antiguo modelo para la creación de nuevos fármacos. Cabe notar que, una de las operaciones bioninformáticas más elementales consiste en la búsqueda de semejanzas entre un fragmento de ADN recién secuenciado y los segmentos ya disponibles de diversos organismos (recuerde y asocie esto con el DPS).

El hallazgo de emparejamientos aproximados permite predecir el tipo de proteína que especificará tal secuencia. Esto no solo proporciona pistas sobre diseños farmacológicos en las etapas iniciales del desarrollo de medicamentos, sino que suprime algunas que constituirán "puzzles" sin resolver. Una serie popular de programas para comparar secuencias de ADN es BLAST (Basic Local Alignment Search Tool) [2] [3] cuyo mecanismo de comparación se aplicó en el desarrollo del nuevo fármaco como se muestra en el esquema de la figura 3.

4. Procedimiento de minería de datos para la inferencia gramatical

La idea considera las experiencias adquiridas [19][1][18], la literatura y las teorías existentes [15], realizando el procesamiento sobre datos que no están estructurados en tablas con atributos diferenciados sino que están codificados

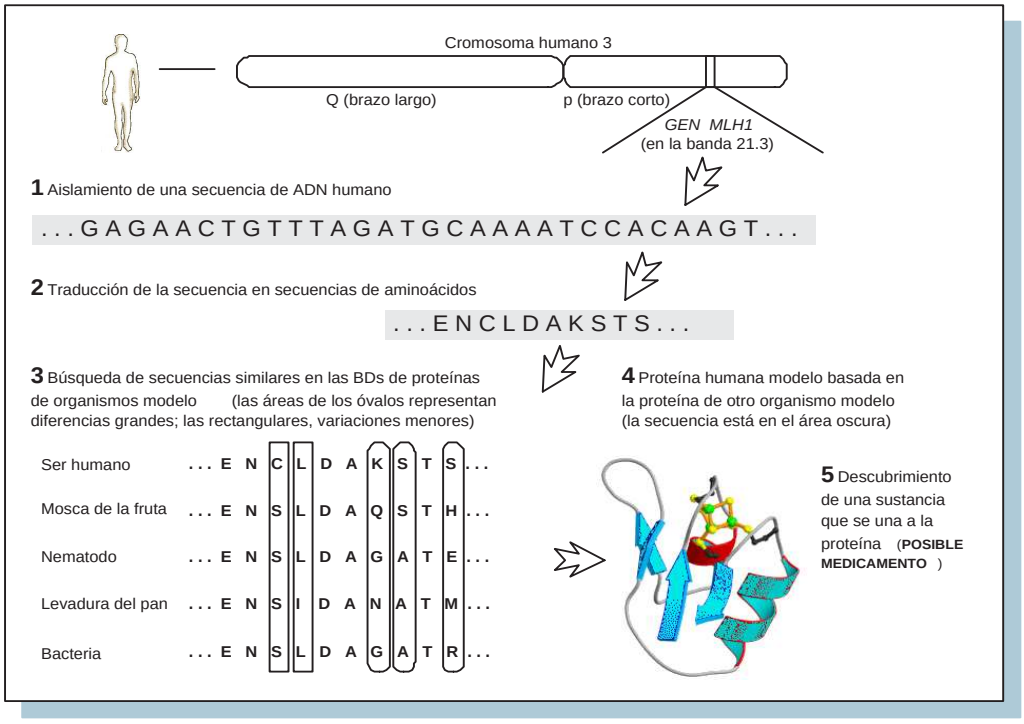

Figura 3: Utilización de la bioinformática en la farmacología (elaborado en base a [12]).

como una sucesión finita de hileras o cadenas.

El procedimiento de minería de datos tiene las siguientes fases:

- Generación de un lenguaje mediante una gramática independiente de contexto. Este lenguaje será la fuente de datos
- Codificación de las cadenas del lenguaje respecto de sus categorías sintácticas
- Prescindiendo de la gramática inicial, descubrimiento de patrones secuenciales sobre el lenguaje codificado. Este descubrimiento denominado "incremental" es una combinación del funcionamiento del DPS y del funcionamiento del alineamiento de secuencias idénticas. Con esto se encontrarán patrones de secuencias que luego serán reeemplazadas por un símbolo identificador
- Reemplazo de las secuencias descubiertas por su identificador. Con lo anterior se almacena el identificador y la secuencia como una regla de producción
- Repetir los dos pasos anteriores hasta que todas las sentencias del lenguaje queden sustituídas por un identificador

4.1. Generación del lenguaje

Consideremos la GIC $\mathcal{G}$ propuesta en [15] sobre la generación de expresiones aritméticas. $\mathcal{G} = (\mathcal{N}, \mathcal{T}, \mathcal{P}, \mathcal{S})$ con $\mathcal{N} = \{\text{Exp, Num, Dig, Op}\}$, $\mathcal{T} = \{0, 1, +, *\}$,

$\mathcal{P}$: Exp $\rightarrow$ Exp Op Exp | (Exp) | Num
Num $\rightarrow$ Dig$^+$
Dig $\rightarrow$ 0 | 1
Op $\rightarrow$ + | *

y $\mathcal{S}$ =Exp.

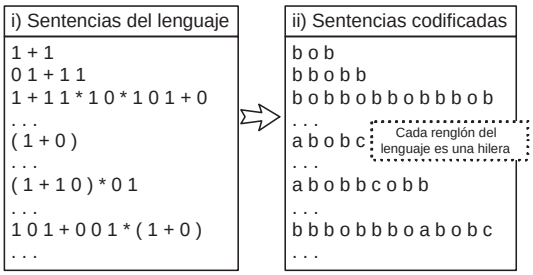

Figura 4: Lenguaje de expresiones aritméticas sobre el cual se infiere su gramática.

Podemos modificar el formalismo de esta GIC de la siguiente forma:

$\mathcal{G} = (\mathcal{N}, \mathcal{T}, \mathcal{P}, \mathcal{S})$ con $\mathcal{N}$ = {E, d, b, o, a, c}, $\mathcal{T} = \{0, 1, +, *, (,)\}$,

$\mathcal{P}$: E → E o E | a E c | n
d → b^+
b → 0 | 1
o → + | *
a → (
c →)

y $\mathcal{S}$ =E, lo que no varía en esencia el carácter de la gramática original.

Con lo anterior, una muestra del lenguaje generado por $\mathcal{G}$ puede observarse en la figura 4 inciso (i). Nótese que cada renglón corresponde a una sentencia aceptada por la gramática.

4.2. Codificación del lenguaje

Considerando el lenguaje que se genera con $\mathcal{G}$, se pueden codificar todos los símbolos de $\mathcal{T}$ con los símbolos de $\mathcal{N}$, para este caso particular sólamente se usan los símbolos {b, o , a, c} como categorías sintácticas. Observar la figura 4, inciso (ii).

4.3. Descubrimiento incremental de patrones secuenciales

El descubrimiento incremental de patrones secuenciales aplicado a lenguajes codificados busca subsecuencias clave en las sentencias del lenguaje. Cada subsecuencia $\mathbf{q}$ tiene un tamaño $\mathbf{w_q}$ que indica el número de símbolos que posee. En este caso particular se fija $1 \leq \mathbf{w_q} \leq 5$ y se define $\mathbf{Q}$ como una hilera de tamaño $\mathbf{w_Q}$. Por convención, en el lenguaje codificado existen muchas hileras que conforman la población del lenguaje. La idea consiste en hallar subsecuencias, identificarlas con un símbolo y reemplazar con ese símbolo las apariciones de las subsecuencias en las hileras de la población, todo lo anterior de forma repetitiva hasta lograr que cada hilera sea identificada por un solo símbolo.

Los pasos detallados son:

1. Para todas las hileras, Mientras $\mathbf{w_Q} > 1$ hacer:

1.1. Para $\mathbf{w_q} = 1.,5$ hacer:

1.1.1. Para todas las hileras hacer:

(a) formar $\mathbf{q}$ a partir de los $\mathbf{w_q}$ primeros símbolos de $\mathbf{Q}$

(b) Calcular la puntuación global $\mathbf{g_q}$ de $\mathbf{q}$ definido como $\mathbf{g_q} = \sum_{i=1}^{cant} \mathbf{p_q^i}$, siendo $\mathbf{p_q} = \frac{\mathbf{w_q} * nro\_apariciones\_q\_en\_Q}{\mathbf{w_Q}}$ la puntuación de $\mathbf{q}$ en $\mathbf{Q}$

1.1.2. Fin para

1.2. Fin para

1.3. Seleccionar la subsecuencia $\mathbf{q}_*$ de mayor puntuación global

1.4. Si $\mathbf{q}_*$ tiene un solo símbolo, entonces reemplazar todas las apariciones consecutivas de ese símbolo por si mismo. Así se crea la regla de producción $\alpha \rightarrow \alpha^+$ (en el caso particular, se reemplazan todos los *dd* por *d*, ver figura 5)

1.5. Si $\mathbf{q}_*$ tiene más de un símbolo, entonces reemplazar todas las apariciones de $\mathbf{q}_*$ en las hileras $\mathbf{Q}$ creando la regla de producción $A \rightarrow contenido\_de(\mathbf{q}_*)$. El símbolo A se genera consecutivamente de forma que la siguiente vez que se crea otra regla de producción, se utiliza $B, C, ...$ y así sucesivamente (ver figura 5).

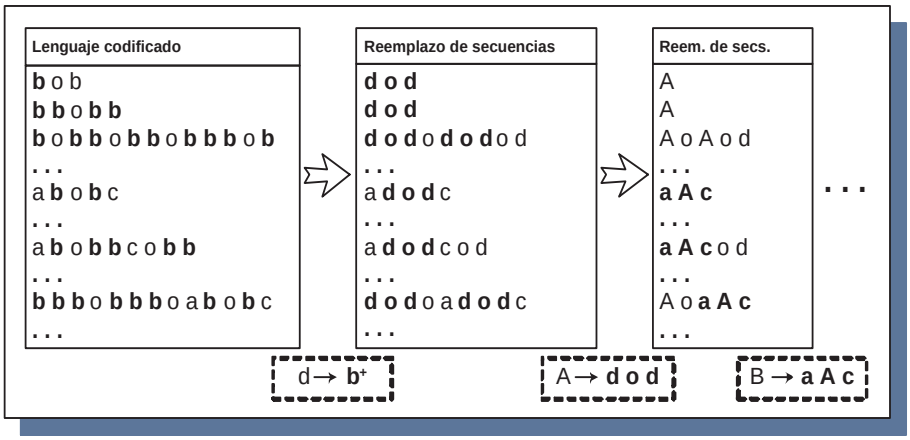

Figura 5: Descubrimiento incremental de patrones secuenciales para lenguajes independientes de contexto.

2. Volver al paso 1 notando que con 1.4 y 1.5 se cambia el tamaño de las hileras de la población del lenguaje

Con el anterior procedimiento se generan reglas de producción que reconocen las sentencias del lenguaje. El número de reglas de producción puede ser considerable por lo que se aplica un método particular de simplificación de gramáticas que obtiene una gramática más compacta.

5. Experimentos

5.1. Semejanza entre reglas

Considerando el lenguaje $\mathcal{L}$ de expresiones aritméticas[1], se aplicó el algoritmo incremental de DPS y se obtuvieron las reglas de producción de la figura 6. Con las partes derechas de las reglas, que constituyen los patrones secuenciales del lenguaje, se calcula una *matriz de sustitución* como se observa en la figura 7, esta matriz muestra los valores de semejanza entre los símbolos terminales. La semejanza entre un par de símbolos consecutivos está relacionada con la frecuencia de aparción de esos símbolos en el lenguaje (es una matriz como BLOSUM [10]). Posteriormente se pueden realizar

[1]El corpus se puede observar en http://www.geocities.com/ramirohp/corpusae.html

Matriz de sustitución

	d	A	C	o	E	B	F	D	.	.	.	P	R
d	23	-1	-2	-3	-4	-5	-6	-7	.	.	.	-21	-22
A	-1	22	-1	-2	-3	-4	-5	-6	.	.	.	-20	-21
C	-2	-1	21	-1	-2	-3	-4	-5	.	.	.	-19	-20
o	-3	-2	-1	20	-1	-2	-3	-4	.	.	.	-18	-19
E	-4	-3	-2	-1	19	-1	-2	-3	.	.	.	-17	-18
B	-5	-4	-3	-2	-1	18	-1	-2	.	.	.	-16	-17
F	-6	-5	-4	-3	-2	-1	17	-1	.	.	.	-15	-16
D	-7	-6	-5	-4	-3	-2	-1	16	.	.	.	-14	-15
.	..	..	..	..	..	..	..	..	.	.	.	..	..
.	..	..	..	..	..	..	..	..	.	.	.	..	..
.	..	..	..	..	..	..	..	..	.	.	.	..	..
P	-21	-20	-19	-18	-17	-16	-15	-14	.	.	.	2	-1
R	-22	-21	-20	-19	-18	-17	-16	-15	.	.	.	-1	1

Figura 7: Matriz de sustitución para las reglas generadas.

alineamientos entre estas secuencias para compactarlas.

En la matriz de sustitución m cada fila i y cada columna j se corresponden con un símbolo no terminal de las reglas de producción generadas. Los símbolos están dispuestos según su frecuencia de aparición, esto es, primero d después A, C, o y así sucesivamente. Para $\mathcal{L}$ se generáron 19 símbolos A, B, ..., S que junto con los símbolos de la codificación d, o, c y a conforman 23 símbolos no terminales (en el

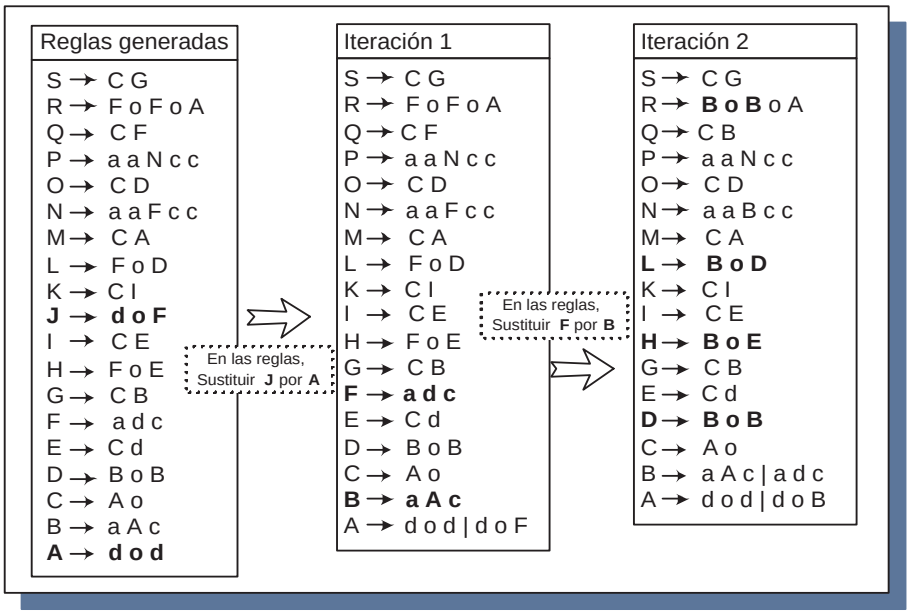

Figura 6: Reglas de producción generadas y algunas iteraciones en su simplificación.

contexto de la bioinformática, los símbolos corresponderían a los aminoácidos). Los valores de la matriz denotan la importancia del alineamiento entre los símbolos no terminales; por ejemplo, $m(d,d) = 23$ denota una grado de similitud alto entre ambos símbolos; $m(d,A)$ denota un grado de similitud de -1.

5.2. Simplificación y compactación de reglas

Con las partes derechas de las reglas de producción (donde las primeras reglas generadas tienen mayor importancia) se buscan *secuencias parecidas* para compactarlas. Los pasos a seguir son:

- Las secuencia β que se puede compactar con la secuencia α se activa con la función de semejanza f; $f(\alpha,\beta) = \begin{cases} 1 & si \ \frac{\sum_{i=1}^{n} m(\alpha_i,\beta_i)}{\sum_{i=1}^{n} m(\alpha_i,\alpha_i)} > \theta; \\ 0 & si \ e.o.c. \end{cases}$
  donde n es la longitud mínima entre las secuencias α y β, θ es un umbral o factor de semejanza con valor 0.4 en este caso particular.
- Las secuencias parecidas se compactan y serán derivadas por un solo símbolo no terminal. El restante símbolo no terminal debe ser reemplazado por el anterior en todas las partes derechas de las reglas.
- Repetir los pasos anteriores hasta que no hayan secuencias parecidas

Por ejemplo, para el lenguaje $\mathcal{L}$ las reglas **dod** y **aAc** no se parecen pues, $f(\mathbf{dod},\mathbf{aAc}) = 0$ ya que $\frac{\sum m(\mathbf{dod},\mathbf{aAc})}{\sum m(\mathbf{dod},\mathbf{dod})} = \frac{-10-2-11}{23+20+23} = \frac{-23}{66} = -0{,}35$ no supera la cantidad 0,40. Sin embargo, las reglas **dod** y **doF** se parecen ya que $\frac{\sum m(\mathbf{dod},\mathbf{doF})}{\sum m(\mathbf{dod},\mathbf{dod})} = \frac{23+20-6}{23+20+23} = \frac{37}{66} = 0{,}56$. Así, las reglas generadas se van simplificando y compactando iterativamente (figuras 6 y 8) hasta tener una gramática $\mathcal{G}' = (\mathcal{N}', \mathcal{T}', \mathcal{P}', \mathcal{S}')$ con $\mathcal{N}' = \{$S, R, E, D, B, A, d, b, o, a, c$\}$, $\mathcal{T}' = \{0, 1, +, *, (,)\}$,

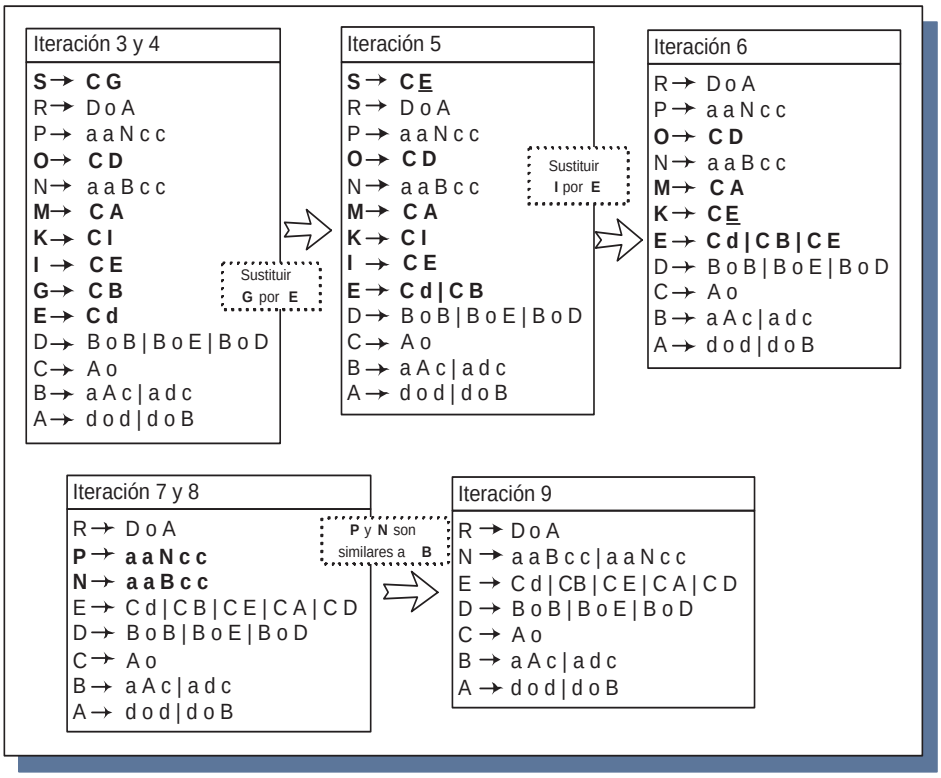

Figura 8: Simplificación y compactación de las reglas de producción generadas.

$\mathcal{P}'$: S → R | E | D | B | A | d
R → DoA
E → Cd | CB | CE | CA | CD
D → BoB | BoE | BoD
C → Ao
B → aAc | adc
A → dod | doB
d → b^+
b → 0 | 1
o → + | *
a → (
c →)
y $\mathcal{S}'$ =S.

6. Conclusiones

En los experimentos, se ha considerado un lenguaje $\mathcal{L}$ generado por una gramática independiente de contexto predeterminada $\mathcal{G}$ y se conocían de antemano las categorías sintácticas **b**, **o**, **a** y **c**; pero luego no se utilizaron ninguna de las propiedades de esa gramática para generar el conjunto de reglas de producción que luego conformaron la gramática $\mathcal{G}'$. La visión se extiende al tratamiento de datos que creemos tengan una estructura gramatical la cual podría ser generada automáticamente. Imaginemos encontrar algo similar para el genoma, para el proteoma o para los lenguajes naturales, la duda está servida.

Referencias

[1] Ramiro Aguilar. *Minería de datos. Fundamentos, técnicas y aplicaciones.* Universidad de Salamanca, 2003.

[2] Stephen F. Altschul, Warren Gish, Webb Miller, Eugene W. Meyers, and David J. Lipman. Basic local alignment search tool. *Molecular Biology*, 215(3):403–410, October 1990.

[3] Stephen F. Altschul, Thomas L. Madden, Alejandro A. Schaffer, Jinghui Zhang, Zheng Zhang, Webb Miller, and David J. Lipman. Gapped BLAST and PSI-BLAST: a new generation of protein database search programs. *Nucleic Acids Research*, 25(17):3389–3402, September 1997.

[4] P. Cabena, P. Hadjinian, R. Stadler, J. Verhees, and A. Zanasi. *Discovering data mining. From concept to implementation.* Prentice Hall, 1998.

[5] L. Corlett. Web content mining: a survey. Technical report, Department of Computer Science, California State University, 2003.

[6] Colin de la Higuera. Current trends in grammatical inference. In *Proceedings of Joint IAPR International Workshops Sspr 2000 and Spr 2000*, pages 130–135. Lecture Notes in Artificial Intelligence, Springer-Verlag, 2002.

[7] Colin de la Higuera. A bibliographical study of grammatical inference. *Pattern Recognition*, 2004.

[8] U. Fayyad, G. Piatetsky-Shapiro, P. Smyth, and R. Uthurusamy. *Advances in Knowledge Discovery and Data Mining.* MIT Press, 1996.

[9] King-Sun Fu. *Syntactic Methods in Pattern Recognition.* Academic Press, 1974.

[10] Steven Henikoff and Jorja G. Henikoff. Amino acid substitution matrices from protein blocks. *Proc. National Academic Science*, 89:10915–10919, 1992.

[11] James Jay Horning. A study of grammatical inference. Technical Report 139, Computer Science Department, Stanford University, 1969.

[12] Ken Howard. La fiebre de la bioinformática. *Investigación y ciencia: nueva genética*, 38:79–82, 2004.

[13] Lillian Lee. Learning of context-free languages: A survey of the literature. Technical Report TR-12-96, Harvard University, 1996.

[14] Vivian López and Ramiro Aguilar. Minería de datos y aprendizaje automático en el procesamiento del lenguaje natural. In *Workshop de minería de datos y aprendizaje, IBERAMIA 2002*, pages 209–216. University of Sevilla, 2002.

[15] Kenneth C. Louden. *Compiler construction. Principles and practice.* International Thomsom Publishing Inc, 1997.

[16] S. Lucas. Structuring chromosomes for context-free grammar evolution. In *IEEECEP: Proceedings of The IEEE Conference on Evolutionary Computation, IEEE World Congress on Computational Intelligence*, pages 130–135, 1994.

[17] L. Miclet. *Structural methods in pattern recognition.* Chapman and Hall, 1986.

[18] Sushmita Mitra and Tinku Acharya. *Data mining. Multimedia, soft computing and bioinformatics.* John Wiley and sons, 2003.

[19] A. Moreno. *Linguística Computacional.* Editorial Síntesis, Madrid, 1998.