Optimización en el Reconocimiento de Patrones*

F.J. Ribadas Pena V.M. Darriba Bilbao M. Vilares Ferro
Departamento de Computación. Universidade de A Coruña.
Campus de Elviña s/n, 15071 A Coruña.
E-mail: ribadas@mail2.udc.es, darriba@mail2.udc.es, vilares@dc.fi.udc.es

Resumen

El objetivo del trabajo es reducir los costes de evaluación de interrogaciones en sistemas de recuperación de información que integren técnicas de reconocimiento de patrones. En este contexto estudiamos la aplicación de estrategias de programación dinámica en situaciones de compartición de estructuras entre patrones.

Nuestra intención es mejorar el rendimiento computacional en esta clase de estrategias, a menudo valoradas por la calidad de los resultados, pero desechadas por su complejidad en favor de enfoques menos sofisticados basados en la simple indexación de términos. El trabajo incluye los primeros resultados experimentales que, aunque preliminares, sugieren su viabilidad.

1 Introducción

Un aspecto crítico de los sistemas de *recuperación de información* (RI), que determina su efectividad, es la indexación. Esto es, la representación de los conceptos de interés en un documento mediante una estructura de datos que permita un posterior proceso de búsqueda. En los primeros sistemas de RI la indexación se llevaba a cabo mediante una selección de las palabras más relevantes que referencian al texto original. Con la reducción en los costes de procesamiento, la tendencia actual es la indexación total de los documentos, donde todas las palabras del texto se consideran potenciales índices. Ello simplifica el proceso, aunque en ocasiones obstaculiza la localización de información relevante para el usuario.

En efecto, las palabras presentes en la consulta no siempre reflejan por si mismas los conceptos relevantes para el usuario. Es su combinación la que proporciona el valor de los conceptos buscados, lo que nos lleva a considerar formalismos de indexación más sofisticados, como las gramáticas de contexto libre [1992]. Ello convierte al reconocimiento de patrones en un modo natural de evaluación de consultas en un sistema de RI. En Smeaton et al. [1994] pueden encontrarse resultados preliminares sobre el uso de estructuras sintácticas y de reconocimiento de patrones en tareas de RI.

En este punto es necesario considerar ciertas limitaciones al uso de patrones en indexación. El lenguaje del documento, a menudo, no podrá estar completamente definido y será frecuente la presencia de ambigüedades. Ello conlleva, por un lado, la conveniencia de considerar todos los posibles análisis en el posterior proceso semántico, reuniendo todos los árboles de análisis en una única estructura que permita compartir las subestructuras comunes, y evitando duplicidades que afectarían a la congruencia de la información. Por otro lado, el reconocimiento de patrones no puede limitarse al tratamiento clásico de modelos exactos, debiendo adaptarse a la consideración de técnicas de reconocimiento aproximado. Nuestro objetivo será aprovechar al máximo la compartición estructural en el proceso de reconocimiento aproximado de patrones, con el objeto de mejorar el rendimiento computacional.

2 Distancia de edición

La similaridad entre dos árboles se basa en el concepto de *distancia de edición*, esto es, el coste de tranformar un árbol en otro aplicando una serie de operaciones de edición. Dados los árboles T_1 y T_2, se define una *operación de edición* como un par $a \to b$, con $a \in \text{etiquetas}(T_1) \cup \{\varepsilon\}$ y $b \in \text{etiquetas}(T_2) \cup \{\varepsilon\}$, donde ε representa la cadena vacía. Se definen tres operaciones de edición: borrado de un nodo de T_1 $(a \to \varepsilon)$, inserción de un nodo en T_2 $(\varepsilon \to b)$ y cambio de etiquetas

* Trabajo parcialmente financiado por los proyectos PGIDT99XI10502B de la Xunta de Galicia y 1FD97-0047-C04-02 Unión Europea.

$(a \rightarrow b)$. Cada una de estas operaciones tiene asociado un coste, $\gamma(a \rightarrow b)$, que se puede extender a una secuencia S de operaciones de edición $s_1,\ s_2,\ \ldots,\ s_n$ en la forma $\gamma(S) = \sum_{i=1}^{|S|}(\gamma(s_i))$. Así, la distancia entre T_1 y T_2 está definida por la métrica :

$$\delta(T_1, T_2) = min\{\gamma(S),\ S \text{ secuencia de operaciones de edición que transforman } T_1 \text{ en } T_2\}$$

Para simplificar el cálculo de $\delta(T_1, T_2)$, Tai [1978] define un tipo especial de secuencias de operaciones de edición, llamado *correspondencia*, que mantiene el orden jerárquico y entre hermanos. Formalmente, dada una ordenación en postorden de los nodos de un árbol T, en la que $T[i]$ se refiere al nodo i-ésimo de T, una *correspondencia* entre T_1 y T_2 es un triple (M, T_1, T_2), donde M es un conjunto de pares de enteros (i, j) verificando, para cada $1 \leq i_1, i_2 \leq |\ T_1\ |$ y $1 \leq j_1, j_2 \leq |\ T_2\ |$:

$i_1 = i_2$ **sii** $j_1 = j_2$
$T_1[i_1]$ a la izq. $T_1[i_2]$ **sii** $T_2[j_1]$ a la izq. $T_2[j_2]$
$T_1[i_1]$ ancestro $T_1[i_2]$ **sii** $T_2[j_1]$ ancestro $T_2[j_2]$

que se corresponde, respectivamente, con una asignación uno a uno, el manteniemiento del orden entre nodos hermanos y el mantenimiento del orden de los ancestros. En la Fig. 1 se presenta un ejemplo de *correspondencia* correcto (derecha) y de una secuencia de operaciones que no constituye una *correspondencia* (izquierda). El coste $\gamma(M)$, de una *correspondencia* (M, T_1, T_2) se calcula a partir de las operaciones de borrado, inserción y cambio de etiquetas en la forma :

$$\gamma(M) = \sum_{(i,j)\in M}\gamma(T_1[i] \rightarrow T_2[j]) + \sum_{i\in\mathcal{D}}\gamma(T_1[i] \rightarrow \varepsilon) + \sum_{j\in\mathcal{I}}\gamma(\varepsilon \rightarrow T_2[j])$$

donde $\mathcal{D}$ y $\mathcal{I}$ son, respectivamente, los nodos de T_1 y T_2 que no participan en M. Tai demuestra que dados los árboles T_1 y T_2:

$$\delta(T_1, T_2) = min\{\gamma(M),\ M \text{ correspondencia de } T_1 \text{ a } T_2\}$$

lo que permite reducir el problema, centrándonos en las secuencias de operaciones que cumplan las condiciones que definen una *correspondencia*.

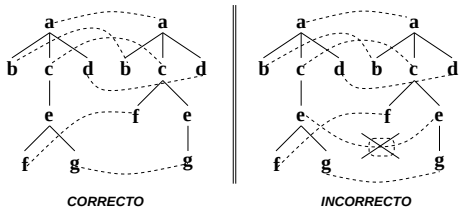


Figura 1: *Ejemplos de correspondencias*

3 El algoritmo de Zhang y Shasha

En este apartado se presentará el algoritmo de reconocimiento aproximado de patrones propuesto por Zhang y Shasha [1989]. Dado un árbol patrón P y un árbol dato D con sus respectivos nodos ordenados en postorden, estos autores presentan un algoritmo ascendente para el cálculo de la distancia entre P y D aplicando técnicas de programación dinámica, aunque sin aplicación directa al caso en el que D fuese un bosque compartido.

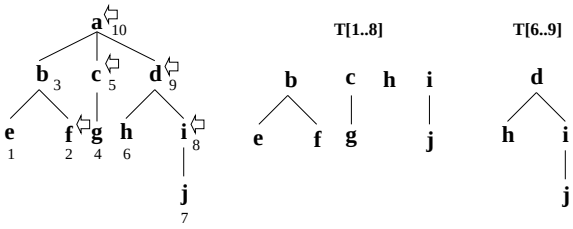


Figura 2: *La distancia entre bosques usando una numeración en postorden*

El algoritmo se basa en el concepto de *l_keyroot* y en el cálculo de distancias inducidas entre bosques. Para un árbol T, el conjunto *l_keyroots*(T) incluye a todos los nodos de T que tengan un hermano a la izquierda así como al nodo raíz de T. Se define $l(i)$ (resp. $anc(i)$) como la hoja más a la izquierda en el subárbol cuya raíz es el nodo T_i (resp. los ancestros de T_i), y se define $T[i..j]$ como la subestructura ordenada inducida comprendida entre los nodos i-ésimo a j-ésimo. En particular, $T[l(i)..i]$ se corresponde con el subárbol cuya raíz es T_i. La Fig. 2 presenta un ejemplo donde se identifican los *l_keyroots* y como se generan los bosques inducidos. Se define la *distancia de edición* entre bosques como una generalización de la distancia entre árboles δ, de la siguiente forma:

$$\text{forest\_dist}(i_1..i_2, j_1..j_2) = \delta(P[i_1..i_2], D[j_1..j_2])$$

El algoritmo identifica los *l_keyroots* en los árboles patrón P y dato D. Para cada par de *l_keyroots* (i, j), con $i \in P$ y $j \in D$, se calculan de forma exhaustiva las distancias entre

todos los posibles pares de bosques inducidos. El cálculo a realizar dependerá de si se están comparando dos árboles o dos bosques de árboles. En el cálculo de *tree_dist*(P, D) para los nodos $i \in anc(s)$ y $j \in anc(t)$ se aplican las fórmulas de la Fig. 3, donde tambien se presenta un gráfico mostrando cómo son los cálculos para árboles y para bosques de árboles.
Finalmente, para el calculo de la distancia entre P y D es suficiente tener en cuenta que

$$\text{tree\_dist}(P, D) = \text{forest\_dist}(l(\text{raiz}(P))..\text{raiz}(P), l(\text{raiz}(D))..\text{raiz}(D))$$

La complejidad temporal de este algoritmo es, en el peor de los casos :

$$\mathcal{O}(|P| \times |D| \times min(\text{depth}(P), \text{leaves}(P)) \times min(\text{depth}(D), \text{leaves}(D)))$$

siendo $|P|$ (resp. $|D|$) el número de nodos en el árbol patrón P (resp. en el árbol dato D), leaves(P) (resp. leaves(D)) el número de hojas en P (resp. en D), y depth(P) (resp. depth(D)) la profundidad de P (resp. de D).

4 *Análisis sintáctico y reconocimiento de patrones*

La consideración de un mecanismo de indexación como el descrito conlleva la integración de un análizador sintáctico que proporcione al sistema de RI los patrones a localizar, a partir del análisis de la interrogación efectuada por el usuario. En este punto es necesario tener en cuenta que ambos aspectos, análisis sintáctico y reconocimiento de patrones, están topológicamente relacionados, en particular en lo referente a los mecanismos que provocan la multiplicación de estructuras en los índices.

Factor determinante es el tipo de representación sintáctica utilizada. Este trabajo se ha desarrollado en el contexto del analizador ICE [1994b], que representa el resultado del análisis mediante un grafo AND-OR. En este grafo, los nodos AND se corresponden con los nodos usuales de un árbol de análisis, mientras que los nodos OR representan las ambigüedades. La compartición se corresponde con árboles completos, pero también con la compartición de una parte de los descendientes de un nodo (Fig. 5).

El tipo de compartición sobre esta parte de la descendencia depende, a su vez, del tipo de estrategia de análisis considerada. De

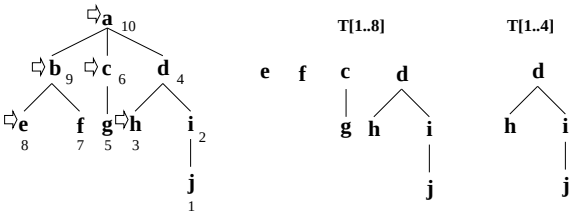


Figura 4: *La distancia entre bosques usando una numeración en postorden inverso*

este modo las técnicas ascendentes provocan la compartición de los constituyentes situados más a la derecha, mientras que las descendentes lo hacen con los situados más a la izquierda, tal y como se muestra en la Fig. 5.

Otra característica interesante de ICE es que los nodos del grafo AND-OR resultante son los elementos mismos de cálculo del analizador, por lo que unidades estructurales y computaciones se identifican en un único concepto que denominamos *item*.

El algorithmo de Zhang y Shasha considera un postorden transversal en la numeración de los nodos, calculando a partir de ella la distancia entre bosques. Ello supone un recorrido en profundidad de las estructuras que, por tanto, requeriría de un analizador sintáctico descendente para evitar la redundancia de cálculos. A su vez ello supone una pérdida de eficacia en relación a las estrategias ascendentes, como es el caso de ICE. En consecuencia se impone una adaptación del algoritmo original para su integración práctica con un analizador sintáctico ascendente.

En este sentido, los nodos en un árbol T serán numerados considerando un postorden transversal inverso, tal y como se muestra en la Fig. 4. Introducimos igualmente *r_keyroots*(T), indicado mediante flechas en la Fig. 4, como el conjunto de los nodos de un árbol T con un hermano a la derecha más la raíz, *raiz*(T), de T. También definimos $r(i)$ como la hoja más a la derecha de las que descienden del subárbol con raíz en T_i.

A partir de aquí, la construcción alternativa para la distancia de edición entre bosques es análoga a la del algoritmo original. Con el objeto de facilitar la comprensión, en la Fig. 6 presentamos los cálculos para árboles y bosques en postorden inverso.
De nuevo, para calcular *tree_dist*(P, D) será suficiente tener en cuenta que

$$\text{tree\_dist}(P, D) = \text{forest\_dist}(r(\text{raiz}(P))..\text{raiz}(P), r(\text{raiz}(D))..\text{raiz}(D))$$

$$\text{forest\_dist}(l(i)..s, l(j)..t) = \begin{cases} \min \left\{ \begin{array}{lcl} \text{forest\_dist}(l(i)..s-1,\ l(j)..t) & + & \gamma(P[s] \to \varepsilon), \\ \text{forest\_dist}(l(i)..s,\ l(j)..t-1) & + & \gamma(\varepsilon \to D[t]), \\ \text{forest\_dist}(l(i)..s-1,\ l(j)..t-1) & + & \gamma(P[s] \to D[t]) \end{array} \right\} \\ \textbf{sii } l(s) = l(i) \textbf{y } l(t) = l(j) \quad \textbf{(árboles)} \\ \min \left\{ \begin{array}{lcl} \text{forest\_dist}(l(i)..s-1,\ l(j)..t) & + & \gamma(P[s] \to \varepsilon), \\ \text{forest\_dist}(l(i)..s,\ l(j)..t-1) & + & \gamma(\varepsilon \to D[t]), \\ \text{forest\_dist}(l(i)..l(s)-1,\ l(j)..l(t)-1) & + & \text{tree\_dist}(s,t) \end{array} \right\} \\ \textbf{en otro caso} \quad \textbf{(bosques)} \end{cases}$$

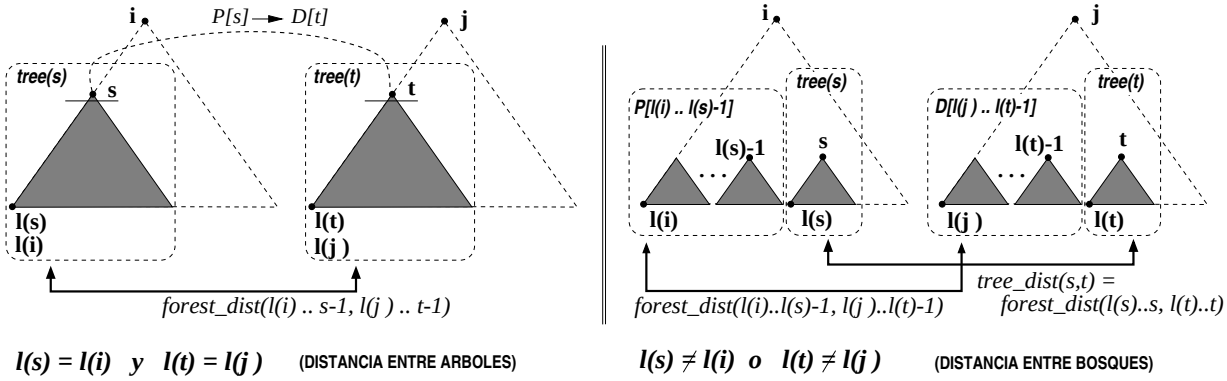


Figura 3: *La distancia entre bosques en el algoritmo de Zhang and Shasha*

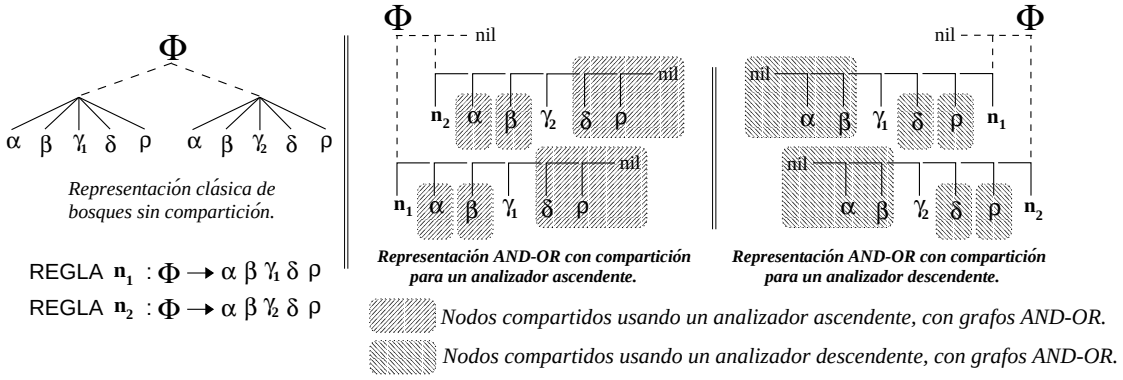


Figura 5: *Cómo los bosques compartidos son construidos usando un formalismo* AND-OR

Los límites espaciales y temporales son los mismos del algoritmo original de Zhang y Shasha.

5 Patrones aproximados y estructuras compartidas

Ofrecemos ahora una explicación simple de como ambos entornos, de análisis sintáctico y de reconocimiento aproximado de patrones, pueden ser integrados de forma eficaz.

Para comenzar, sea P un árbol ordenado y etiquetado, y sea D un grafo AND-OR, ambos construidos en nuestro entorno de análisis sintáctico ascendente. Identificaremos a P con una interrogación y a D con una parte de la representación sintáctica de una base de datos textual con cierto grado de ambigüedad. Presentaremos la manera en la que calculamos la distancia entre un árbol patrón y el conjunto de árboles representados en el grafo AND-OR, así como la forma de mejorar el rendimiento computacional sobre la base de la compartición de estructuras presentes en el resultado del análisis sintáctico. La complejidad temporal del algoritmo será ahora, en el peor de los casos:

$$\mathcal{O}(|P| \times |D| \times min(\text{depth}(P), \text{leaves}(P)) \times min(\text{depth}(D), \text{leaves}(D)))$$

siendo $|D|$ el número máximo de nodos de un árbol en D, $\text{leaves}(D)$ el número máximo de hojas para un árbol en D, y $\text{depth}(D)$ la profundidad máxima de un árbol en D.

Sea $P[s]$ el nodo actual en el postorden in-

$$\text{forest\_dist}(r(i)..s, r(j)..t) = \begin{cases} \min \left\{ \begin{array}{lcl} \text{forest\_dist}(r(i)..s-1,\ r(j)..t) & + & \gamma(P[s] \to \varepsilon), \\ \text{forest\_dist}(r(i)..s,\ r(j)..t-1) & + & \gamma(\varepsilon \to D[t]), \\ \text{forest\_dist}(r(i)..s-1,\ r(j)..t-1) & + & \gamma(P[s] \to D[t]) \end{array} \right\} \\ \textbf{sii } r(s) = r(i) \textbf{ y } r(t) = r(j) \\ \\ \min \left\{ \begin{array}{lcl} \text{forest\_dist}(r(i)..s-1,\ r(j)..t) & + & \gamma(P[s] \to \varepsilon), \\ \text{forest\_dist}(r(i)..s,\ r(j)..t-1) & + & \gamma(\varepsilon \to D[t]), \\ \text{forest\_dist}(r(i)..r(s)-1,\ r(j)..r(t)-1) & + & \text{tree\_dist}(s,t) \end{array} \right\} \\ \textbf{en otro caso} \end{cases}$$

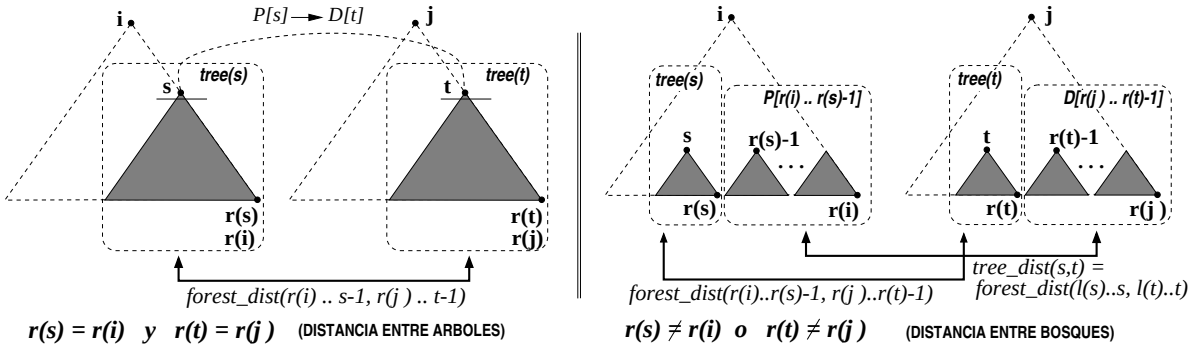


Figura 6: *La distancia entre bosques en nuestra propuesta*

verso de P e $i \in \text{anc}(s)$ un *r_keyroot*. Dado un nodo OR en $D[k]$ podemos distinguir dos casos, dependiendo de la situación de dicho nodo OR en relación a los *r_keyroots* de D.

5.1 Compartiendo en un mismo r_keyroot

Sean $D[t']$ y $D[t'']$ los nodos tratados en paralelo sobre dos ramas etiquetadas $D[k']$ y $D[k'']$ en el nodo OR $D[k]$. Tenemos que $j \in anc(t') \cap anc(t'')$, esto es, el subárbol con raíz en el *r_keyroot* $D[j]$ incluye las alternativas OR $D[k']$ y $D[k'']$.

Tal situación es mostrada en el diagrama superior de la Fig. 7, usando una representación clásica y un grafo AND-OR. Aquí, la parte suavemente sombreada se refiere a los nodos cuyas distancias han sido calculadas en el postorden inverso antes del nodo OR $D[k]$. La parte fuertemente sombreada representa una estructura compartida. La notación "• • •" en los nodos AND-OR expresa el descenso a lo largo de la rama más a la derecha del correspondiente árbol.

Asumiremos que los nodos $D[r(t') - 1]$ y $D[r(t'') - 1]$ son el mismo, esto es, sus correspondientes subárboles son compartidos. Así, $D[r(t')]$ (resp. $D[r(t'')]$) es el siguiente nodo en $D[k']$ (resp. $D[k'']$) a ser tratado, una vez la distancia sobre la estructura compartida haya sido calculada.

En este punto, nuestro objetivo es calcular los valores para $\textit{forest\_dist}(r(i)..s,\ r(j)..\hat{t})$, $\hat{t} \in \{t',\ t''\}$, probando que podemos traducir la compartición de estructuras durante el análisis sintáctico en compartición de cálculos para estas distancias.
Formalmente, los valores para $\textit{forest\_dist}(r(i)..s,\ r(j)..\hat{t})$, $\hat{t} \in \{t',\ t''\}$ vienen dados por las fórmulas de la Fig. 7.

En este caso tenemos que $r(j) \neq r(\hat{t})$, puesto que asumimos que existe una compartición entre $P[r(\hat{t})]$ y $P[r(j)]$. Así, podemos centrarnos en el cálculo alternativo, donde:

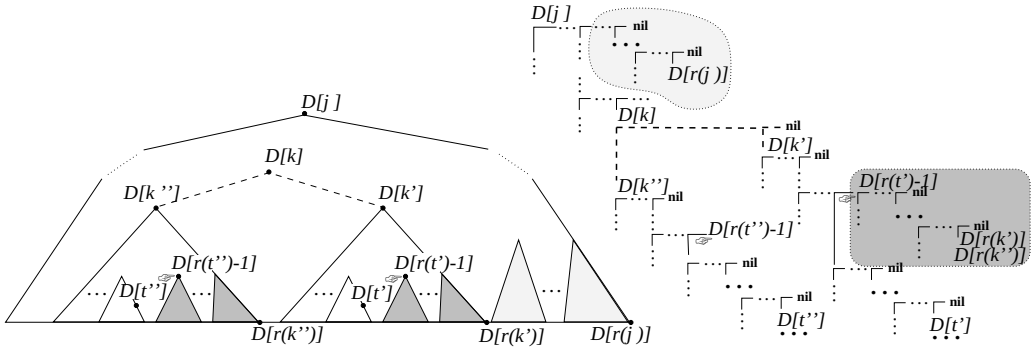
1. Los valores $\textit{forest\_dist}(r(i)..s\text{-}1,\ r(j)..\hat{t})$, $\hat{t} \in \{t',\ t''\}$ han sido calculados por el algoritmo en un paso previo. De este modo, la compartición proporcionada por el analizador sintáctico no tiene consecuencias en el cálculo de las distancias.

2. Son posibles dos casos en relación a la naturaleza de los nodos $D[\hat{t}]$, $\hat{t} \in \{t',\ t''\}$, estos son:

   • Si ambos nodos son hojas, entonces $r(\hat{t}) = \hat{t}$. En consecuencia tenemos que

   $$\begin{array}{rl} D[t'-1] = & D[r(t')-1] = \\ & D[r(t'')-1] = \quad D[t''-1] \end{array}$$

   y por lo tanto para $\hat{t} \in \{t',\ t''\}$ los valores $\textit{forest\_dist}(r(i)..s, r(j)..\hat{t}\text{-}1)$, son también los mismos.



$$\text{forest\_dist}(r(i)..s, r(j)..\hat{t}) = \begin{cases} \min \left\{ \begin{array}{lcl} \text{forest\_dist}(r(i)..s-1,\ r(j)..\hat{t}) & + & \gamma(P[s] \to \varepsilon), \\ \text{forest\_dist}(r(i)..s,\ r(j)..\hat{t}-1) & + & \gamma(\varepsilon \to D[\hat{t}]), \\ \text{forest\_dist}(r(i)..s-1,\ r(j)..\hat{t}-1) & + & \gamma(P[s] \to D[\hat{t}]) \end{array} \right\} \\ \textbf{sii } r(s) = r(i) \textbf{ y } r(\hat{t}) = r(j) \\ \\ \min \left\{ \begin{array}{lcl} \text{forest\_dist}(r(i)..s-1,\ r(j)..\hat{t}) & + & \gamma(P[s] \to \varepsilon), \\ \text{forest\_dist}(r(i)..s,\ r(j)..\hat{t}-1) & + & \gamma(\varepsilon \to D[\hat{t}]), \\ \text{forest\_dist}(r(i)..r(s)-1,\ r(j)..r(\hat{t})-1) & + & \text{tree\_dist}(s, \hat{t}) \end{array} \right\} \\ \textbf{en otro caso} \end{cases}$$

Figura 7: *Compartiendo en un mismo* r_keyroot

- De otro modo, siguiendo el postorden inverso, llegaríamos a las hojas más a la derecha en la derivación de $D[t']$ y $D[t'']$, donde podríamos aplicar el razonamiento del caso anterior.

3. Los valores para las distancias *forest_dist*$(r(i)..r(s)\text{-}1,\ r(j)..r(\hat{t})\text{-}1)$, $\hat{t} \in \{t',\ t''\}$ son idénticos, dado que los nodos $D[r(\hat{t})\text{-}1]$, $\hat{t} \in \{t',\ t''\}$ son compartidos por el analizador sintáctico.

5.2 Compartiendo entre diferentes r_keyroots

Tenemos que $j' \in anc(t')$ y $j'' \in anc(t'')$, donde $j' \neq j''$, son dos *r_keyroots*. Tenemos también un nodo OR $D[k]$ que es ancestro común de esos dos nodos. Suponemos que los *r_keyroots* se encuentran en ramas de análisis diferentes, esto es, que existe un *r_keyroot*, $D[j']$ (resp. $D[j'']$), en la rama etiquetada $D[k']$ (resp. $D[k'']$).

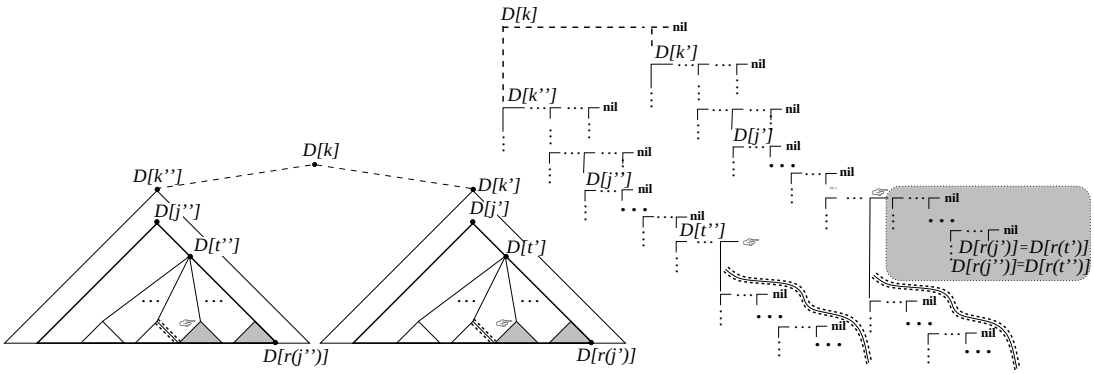
Nuestro objetivo será calcular los valores para las distancias *forest_dist*$(r(i)..s,\ r(\hat{\jmath})..\hat{t})$, donde los pares $(\hat{\jmath}, \hat{t})$ están en $\{(j',\ t'),\ (j'', t''\ )\}$. Estos valores vienen dados por las fórmulas de la Fig. 8.

Esta situación, mostrada en el diagrama superior de la Fig. 8, hace posible $r(s) = r(i)$ y $r(\hat{t}) = r(\hat{\jmath})$. En este primer caso, podemos asumir que los constituyentes más a la derecha son compartidos por los nodos $D[\hat{t}]$, $\hat{t} \in \{t',\ t''\}$. Podemos también asumir que estos constituyentes son un subconjunto propio de la descendencia de dichos nodos dado que, de otro modo, nuestro analizador garantizaría que $D[\hat{t}]$, $\hat{t} \in \{t',\ t''\}$ fuesen también compartidos.

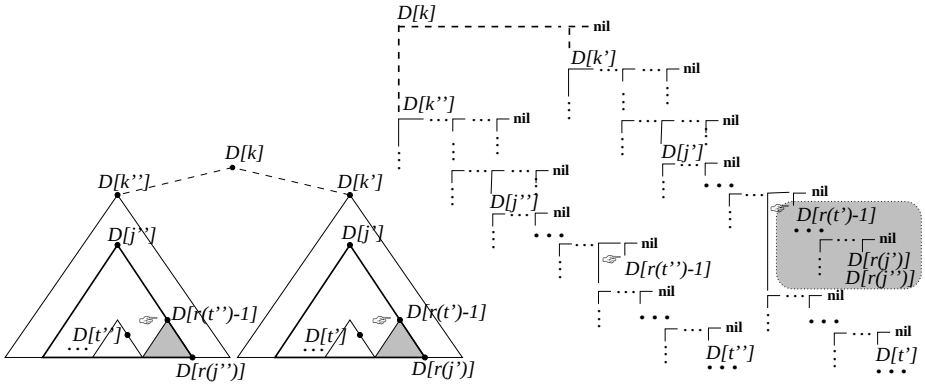
Teniendo en cuenta la estrategia de análisis sintáctico que hemos adoptado, que identifica estructuras sintácticas y cálculos, concluimos que las distancias *forest_dist*$(r(i)..s,\ r(\hat{\jmath})..\hat{t})$, con $(\hat{\jmath}, \hat{t}) \in \{(j',\ t'),\ (j'', t''\ )\}$, no dependen de cálculos previos sobre la parte compartida, tal y como se muestra en la parte izquierda del primer caso de la Fig. 8. Así, esta compartición no tiene consecuencias sobre los cálculos, aunque si los tendrá en el cálculo de las distancias para nodos en la rama más a la derecha del árbol inmediatamente a la izquierda de los constituyentes compartidos, lo que denotamos mediante una línea doblemente punteada en el diagrama.

Consideramos ahora el segundo caso, esto es, el cálculo de la distancia entre bosques cuando $r(\hat{\jmath}) \neq r(\hat{t})$, tal y como aparece en el diagrama inferior de la Fig. 8. Así, en relación a los valores alternativos posibles para el cálculo del mínimo, tenemos que:

1. Los valores *forest_dist*$(r(i)..s\text{-}1, r(\hat{\jmath})..\hat{t})$, $(\hat{\jmath}, \hat{t}) \in \{(j',\ t'),\ (j'', t''\ )\}$ ya han sido calculados en un paso previo del algoritmo, y la compartición sintáctica no in-



PRIMER CASO



SEGUNDO CASO

$$\text{forest\_dist}(r(i)..s, r(\hat{\jmath})..\hat{t}) = \begin{cases} \min \left\{ \begin{array}{lcl} \text{forest\_dist}(r(i)..s-1,\ r(\hat{\jmath})..\hat{t}) & + & \gamma(P[s] \to \varepsilon), \\ \text{forest\_dist}(r(i)..s,\ r(\hat{\jmath})..\hat{t}-1) & + & \gamma(\varepsilon \to D[\hat{t}]), \\ \text{forest\_dist}(r(i)..s-1,\ r(\hat{\jmath})..\hat{t}-1) & + & \gamma(P[s] \to D[\hat{t}]) \end{array} \right\} \\ \textbf{sii } r(s) = r(i) \textbf{ y } r(\hat{t}) = r(\hat{\jmath}) \\ \min \left\{ \begin{array}{lcl} \text{forest\_dist}(r(i)..s-1,\ r(\hat{\jmath})..\hat{t}) & + & \gamma(P[s] \to \varepsilon), \\ \text{forest\_dist}(r(i)..s,\ r(\hat{\jmath})..\hat{t}-1) & + & \gamma(\varepsilon \to D[\hat{t}]), \\ \text{forest\_dist}(r(i)..r(s)-1,\ r(\hat{\jmath})..r(\hat{t})-1) & + & \text{tree\_dist}(s, \hat{t}) \end{array} \right\} \\ \textbf{en otro caso} \end{cases}$$

Figura 8: *Compartiendo entre diferentes* r_keyroots

fluye en el cálculo de las distancias.

2. Distinguimos dos casos en relación a la naturaleza de los nodos $D[\hat{t}]$, con $\hat{t} \in \{t',\ t''\}$. Aplicaremos el mismo razonamiento considerado cuando sólo teníamos un *r_keyroot*:

   • Si ambos nodos son hojas, entonces $r(\hat{t}) = \hat{t}$. En consecuencia tenemos que

   $$D[t'-1] = \quad D[r(t')-1] = \\ D[r(t'')-1] = \quad D[t''-1]$$

   y para $\hat{t} \in \{t',\ t''\}$ las distancias *forest_dist*$(r(i)..s, r(j)..\hat{t}\text{-}1)$ son iguales.

   • De otro modo, siguiendo el postorden inverso, alcanzamos las hojas más a la derecha en las derivaciones de $D[t']$ y $D[t'']$, donde se aplicaría el razonamiento del caso anterior.

3. Los valores para las distancias *forest_dist*$(r(i)..r(s)\text{-}1,\ r(\hat{j})..r(\hat{t})\text{-}1)$, con $\hat{t} \in \{t',\ t''\}$ son idénticos, dado que los árboles con raíces en $D[r(\hat{t})\text{-}1]$, $\hat{t} \in \{t',\ t''\}$ son compartidos por el analizador sintáctico.

6 *Resultados experimentales*

Consideramos el lenguaje de las expresiones aritméticas para ilustrar la discusión, comparando nuestra propuesta con Tai [1978], y Zhang y Shasha [1989]. Consideramos dos gramáticas deterministas, $\mathcal{G}_L$ y $\mathcal{G}_R$, representando las versiones asociativas por la izquierda y por la derecha de los operadores

aritméticos; y una no determinista $\mathcal{G}_N$. Asumimos que los analizadores han sido generados mediante ICE [1994b], y que los tests han sido aplicados sobre datos de la forma $a_1 + a_2 + \ldots + a_i + a_{i+1}$, con i impar. En el caso no determinista estos programas tienen un número, C_i, de análisis ambiguos que crece exponencialmente con i:

$$C_0 = C_1 = 1 \quad \text{y} \quad C_i = \left(\begin{array}{c} 2i \\ i \end{array} \right) \frac{1}{i+1}, \text{ if } i > 1$$

Como patrón hemos considerado los árboles deterministas generados por entradas de la forma $a_1 + b_1 + a_3 + b_3 + \ldots + b_{i-1} + a_{i-1} + b_{i+1} + a_{i+1}$, donde $b_j \neq a_{j-1}$.

En el caso determinista, los patrones han sido construidos a partir de la interpretación asociativa por la izquierda (resp. asociativa por la derecha) de $\mathcal{G}_L$ (resp. $\mathcal{G}_R$), lo que nos permite evaluar el impacto de la orientación de los árboles en el rendimiento. Así, la Fig. 9 prueba la adaptación de nuestra propuesta (resp. del algoritmo de Zhang y Shasha) a las derivaciones recursivas por la izquierda (resp. por la derecha), lo cual corrobora nuestras conclusiones. Estos tests también muestran la independencia del algoritmo de Tai de la topología exhibida por las reglas gramaticales. Ello se debe a que la propuesta de Tai es un algoritmo descendente que no utiliza el concepto de *key_root* y su complejidad no depende de la disposición de los nodos.

En el caso no determinista, los patrones son construidos sobre la interpretación asociativa por la izquierda de la pregunta, lo cual no es relevante puesto que las reglas en $\mathcal{G}_N$ son simétricas. En este caso evaluamos la ganancia en eficacia computacional debida a la compartición de cálculos en programación dinámica, como se muestra en la Fig. 9.

7 Conclusiones

Cualquier acercamiento práctico a la implementación de sistemas RI está basado en dos premisas: una representación común de documentos e interrogaciones, y la consideración de un conjunto de hipótesis de simplificación para las funciones de recuperación. En el primer caso nuestro trabajo propone un mecanismo que parece, a juzgar por los resultados preliminares, mejorar las prestaciones computacionales en técnicas de recuperación de información basadas en el uso de reconocimiento de patrones. En el segundo, la consideración de tecnología específica para

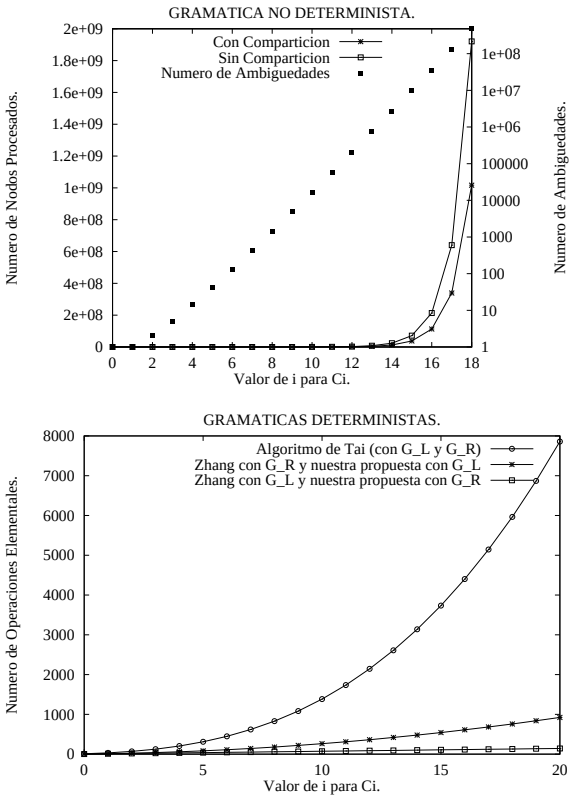


Figura 9: *Resultados experimentales*

la localización de patrones aproximados permite relajar las especificaciones sobre las funciones de recuperación.

Referencias

[1992] Kilpelainen, P., Mannila, H.: *Grammatical Tree Matching.* Lecture Notes in Computer Science **644** (1992) 159–171.

[1994] Smeaton, A.F and O'Donell, R. and Kelley,F.: *Indexing Structures Derived from Syntax in TREC-3: System Description.* Proc. of 3^{rd} Text REtrieval Conference (TREC-3), NIST Special Publication.

[1978] Tai, Kuo-Chung.: *Syntactic error correction in programming languages.* IEEE Transactions on Software Engineering **4(5)** (1978) 414–425.

[1994b] Vilares, M., Dion, B.A.: *Efficient incremental parsing for context-free languages.* Proc. of the 5^{th} IEEE International Conference on Computer Languages (1994) 241–252, Toulouse, France.

[1989] Zhang, K., Shasha, D.: *Simple fast algorithms for the editing distance between trees and related problems.* SIAM Journal on Computing (1989) **18** 1245–1262.