Approximate VLDC Pattern Matching in Shared-Forest

M. Vilares[1], F.J. Ribadas[2], and V.M. Darriba[2]

[1] Departamento de Computación, Universidad de La Coruña
Campus de Elviña s/n, 15071 La Coruña, Spain
vilares@udc.es
http://coleweb.dc.fi.udc.es/

[2] Escuela Superior de Ingeniería Informática, Universidad de Vigo
Edificio Politécnico, As Lagoas, 32004 Orense, Spain
ribadas@uvigo.es, darriba@uvigo.es

© Springer-Verlag

Abstract. We present a matching-based proposal intended to deal with querying for structured text databases. Our approach extends approximate VLDC matching techniques by allowing a query to exploit sharing of common parts between patterns used to index the document.

1 Introduction

Any effort in the development of effective retrieval methods is based on finding a related representation for documents and requests. The main objective is to determine a limited number of index terms for the major concepts in the document.

Until recently, indexing was accomplished by creating a bibliographic citation in a structured file that references the original text. This approach allows a reduction in time and space bounds, although the ability to find information on a particular subject is limited by the system which creates index terms for that subject. At present, the significant reduction in cost processing has propitiated the notion of total document indexing, for which all words in a document are potential index descriptors. This reduction saves the indexer from entering index terms that are identical to words in the document, but does not facilitate the finding of relevant information for the user.

In effect, the words used in a query do not always reflect the value of the concepts being presented. It is the combination of these words and their semantic implications that contain the value of these concepts which leads us to more sophisticated index representations, such as syntactic structures and context-free grammars [2] [3]. This information is inherent in the document and query. So, matching becomes a possible mechanism for extracting a common pattern from multiple data and we could use it to locate information of linguistic interest in

natural language processing. Some related work about using syntactic structures and matching techniques in IR can be found in Smeaton et al. [4] [5] [6].

However, the language intended to represent the document can often only be approximately defined, and therefore ambiguity arises. Since it is desirable to consider all possible parses for semantic processing, it is convenient to merge parse trees as much as possible into a single structure that allows them to share common parts. Although in the case of the query, language ambiguity could probably be eliminated, queries could vary widely from indexes or some structural details are unknown, and an approximate matching strategy in the presence of variable length don't cares (VLDC) becomes necessary [10].

Previous works do not provide a mechanism to fully exploit structural sharing in VLDC matching. Our aim is to cover the lack of proposals in this domain.

2 The editing distance

Given P, a pattern tree, and D, a data tree, we define an *edit operation* as a pair $a \to b$, $a \in \text{labels}(P) \cup \{\varepsilon\}$, $b \in \text{labels}(D) \cup \{\varepsilon\}$, $(a, b) \neq (\varepsilon, \varepsilon)$, where ε represents the empty string. We can delete a node $(a \to \varepsilon)$, insert a node $(\varepsilon \to b)$, and change a node $(a \to b)$. Each edit operation has an associated cost, $\gamma(a \to b)$, that we extend to a sequence S of edit operations $s_1, s_2, \ldots, s_n$ in the form $\gamma(S) = \sum_{i=1}^{|S|}(\gamma(s_i))$. The distance between P and D is defined by the metric:

$$\delta(P, D) = \min\{\gamma(S),\ S \text{ editing sequence taking } P \textit{ to } D\}$$

Given an inverse postorder traversal, as it is shown in Fig. 1, to name each node i of a tree T by $T[i]$, a *mapping* from P to D is a triple (M, P, D), where M is a set of integer pairs (i, j) satisfying, for each $1 \leq i_1, i_2 \leq \mid P \mid$ and $1 \leq j_1, j_2 \leq \mid D \mid$:

$i_1 = i_2$	**iff** $j_1 = j_2$
$P[i_1]$ is to the left of $P[i_2]$	**iff** $D[j_1]$ is to the left of $D[j_2]$
$P[i_1]$ is an ancestor of $P[i_2]$	**iff** $D[j_1]$ is an ancestor of $D[j_2]$

which corresponds, in each case, to one-to-one assignation, sibling order preservation and ancestor order preservation. The cost, $\gamma(M)$, of a mapping (M, P, D) is computed from relabeling, deleting and inserting operations, as follows:

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

where $\mathcal{D}$ and $\mathcal{I}$ are, respectively, the nodes in P and D not touched by any line in M. Tai proves, given trees P and D, that

$$\delta(P, D) = min\{\gamma(M),\ M \text{ mapping from } P \text{ to } D\}$$

which allows us to focus on edit sequences being a mapping. We show in Fig. 2 one example of mapping between two trees, and a sequence of edit operations

not constituting a mapping. We also introduce *r_keyroots*(T) as the set of all nodes in a tree T which have a right sibling plus the root, *root*(T), of T. We shall proceed through the nodes determining mappings from all leaf r_keyroots first, then all r_keyroots at the next higher level, and so on to the root. The set of *r_keyroots*(T) is indicated by arrows in Fig. 1.

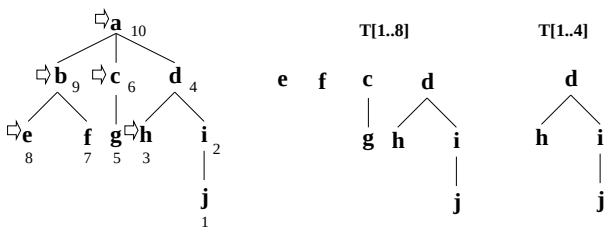


Fig. 1. The forest distance using an inverse postorder numbering

In dealing with approximate VLDC pattern matching, some structural details can be omitted in the target tree and different strategies are then applicable. Following Zhang *et al.* in [10] we introduce two different definitions to VLDC matching:

- The VLDC substitutes for part of a path from the root to a leaf of the data tree. We represent such a substitution, shown in Fig. 2, by a vertical bar "|", and call it a *path*-VLDC.
- The VLDC matches part of such a path and all the subtrees emanating from the nodes of that path, except possibly at the lowest node of that path. At the lowest node, the VLDC symbol can substitute for a set of leftmost subtrees and a set of rightmost subtrees. We call this an *umbrella*-VLDC, and represent it by a circumflex "$\wedge$", as shown in Fig. 2.

To formalize the consideration of pattern trees with VLDC's requires the capture of the notion of VLDC-*substitution* for nodes in the target tree P labeled $|$ or $\wedge$ as previously introduced. So, given a data tree D and a substitution s on P, we redefine:

$$\delta(P, D) = \min_{s \in \mathcal{S}}\{\delta(P, D, s)\}$$

where $\mathcal{S}$ is the set of all possible VLDC-substitutions, and $\delta(P, D, s)$ is the distance $\delta(\bar{P}, D)$, being $\bar{P}$ the result of apply the substitution s to P. As a consequence, no cost is induced by VLDC-substitutions.

3 Approximate VLDC tree matching

The major question of Zhang *et al.* in [9] [10], is the tree distance algorithm itself. However, parsing and tree-to-tree correction are topologically related and,

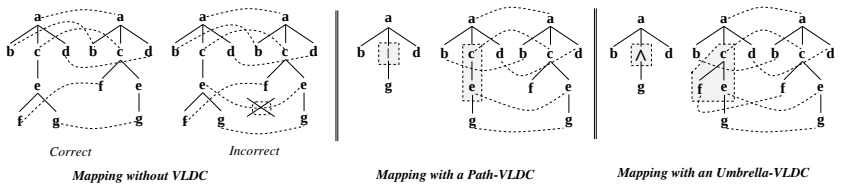


Fig. 2. An example on mappings

to get the best performance, it is necessary to understand the mechanisms that cause the phenomenon of tree duplication.

A major factor to take into account is the syntactic representation used. We choose to work in the parsing context described for ICE [8]. Here, authors represent a parse as the chain of the context-free rules used in a leftmost reduction of the input sentence, rather than as a tree. When the sentence has distinct parses, the set of all possible parse chains is represented in finite shared form by a context-free grammar that generates that possibly infinite set.

This difference with most other parsers is only apparent, since context-free grammars can be represented by AND-OR graphs that in our case are precisely the shared-forest graph [1]. In this graph, AND-nodes correspond to the usual parse-tree nodes, while OR-nodes correspond to ambiguities. Sharing of structures is represented by nodes accessed by more than one other node and it may correspond to sharing of a complete subtree, but also sharing of a part of the descendants of a given node. This allows us to gain in sharing efficiency in relation to classic representations, as shown in Fig. 3.

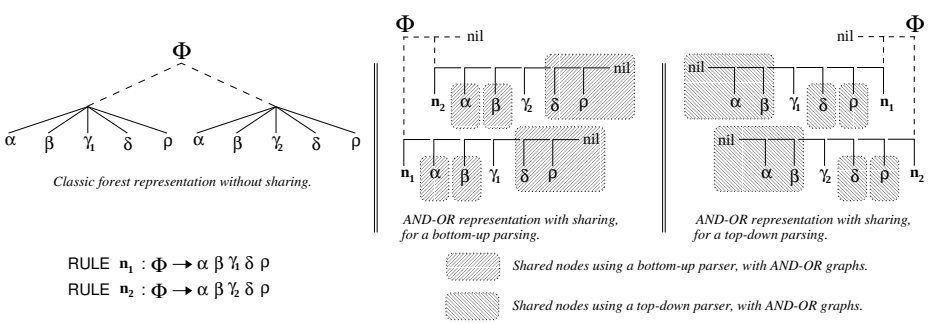


Fig. 3. How shared forest are built using an AND-OR formalism

It is also important the parsing scheme applied. So, bottom-up parsing may share only the rightmost constituents, while top-down parsing may only share the leftmost ones. This relies to the type of search used to built the forest. Breadth first search results on bottom-up constructions and depth first search results on top-down ones, as it is also shown in Fig. 3.

At this level, one major observation we noted is that Zhang *et al.* consider a postorder traversal, computing the forest distance by left-recursion on this search. As a consequence, we would need to consider a top-down parsing architecture to avoid redundant computations. However, top-down parsers are not computationally efficient, and a bottom-up approach, as is the case of ICE, requires a rightmost search of tree constituents. This implies redefining the architecture of the original matching strategy.

To accomplish this change, we introduce $r(i)$ (resp. $\mathrm{anc}(i)$) as the rightmost leaf descendent of the subtree rooted at $T[i]$ (resp. the ancestors of $T[i]$) in a tree T, and $T[i..j]$ as the ordered sub-forest of T induced by the nodes numbered i to j inclusive, as it is shown in Fig. 1. In particular, we have $T[r(i)..i]$ is the tree rooted at $T[i]$. We now define the *forest edition distance* between a target tree P and a data tree D, as a generalization of δ, in the form

$$\mathrm{f_d}(P[s_1..s_2], D[t_1..t_2]) = \delta(P[s_1..s_2], D[t_1..t_2])$$

that we shall denote $\mathrm{f_d}(s_1..s_2, t_1..t_2)$ when the context is clear. Intuitively, this concept computes the distance between two nodes, $P[s_2]$ and $D[t_2]$, in the context of their left siblings in the corresponding trees, while the tree distance, $\delta(P[s_2], D[t_2])$, is computed only from their descendants.

To be precise, given a pattern tree P and a data tree D, we can compute the editing distance $\mathrm{t_d}(P, D)$ applying the formulae that follow [7], for nodes $i \in \mathrm{anc}(s)$ and $j \in \mathrm{anc}(t)$, assuming $P[s]$ is not an incomplete structure:

$$\mathrm{f_d}(r(i)..s, r(j)..t) = \begin{cases} \min \left\{ \begin{array}{ll} \mathrm{f_d}\ (r(i)..s-1,\ r(j)..t) & +\ \gamma(P[s] \to \varepsilon), \\ \mathrm{f_d}\ (r(i)..s,\ r(j)..t-1) & +\ \gamma(\varepsilon \to D[t]), \\ \mathrm{f_d}\ (r(i)..s-1,\ r(j)..t-1) & +\ \gamma(P[s] \to D[t]) \end{array} \right\} \\ \textbf{iff}\ r(s) = r(i)\ \textbf{and}\ r(t) = r(j) \\ \\ \min \left\{ \begin{array}{ll} \mathrm{f_d}\ (r(i)..s-1,\ r(j)..t) & +\ \gamma(P[s] \to \varepsilon), \\ \mathrm{f_d}\ (r(i)..s,\ r(j)..t-1) & +\ \gamma(\varepsilon \to D[t]), \\ \mathrm{f_d}\ (r(i)..r(s)-1,\ r(j)..r(t)-1) & +\ \mathrm{t_d}(s,t) \end{array} \right\} \\ \textbf{otherwise} \end{cases}$$

When $P[s]$ is either "|" or "$\wedge$" formulae must be adapted and the process is illustrated in Fig. 4. We first assume $P[s]$ is "|":

$$\mathrm{f_d}(r(i)..s, r(j)..t) = \min \left\{ \begin{array}{ll} \mathrm{f_d}(r(i)..s-1,\ r(j)..t) & +\ \gamma(P[s] \to \varepsilon), \\ \mathrm{f_d}(r(i)..s,\ r(j)..t-1) & +\ \gamma(\varepsilon \to D[t]), \\ \mathrm{f_d}(r(i)..s-1,\ r(j)..t-1) & +\ \gamma(P[s] \to D[t]), \\ \mathrm{f_d}(\phi, D[r(j)]..t-1) & +\ \min_{t_k}\{\mathrm{t_d}(s,t_k) - \\ & \quad \mathrm{t_d}(\phi,t_k)\} \quad 1 \le k \le n_t \end{array} \right\}$$

For the case where $P[s]$ is "$\wedge$", formulae are the following:

$$\mathrm{f_d}(r(i)..s, r(j)..t) = \min \left\{ \begin{array}{ll} \mathrm{f_d}(r(i)..s-1,\ r(j)..t) & +\ \gamma(P[s] \to \varepsilon), \\ \mathrm{f_d}(r(i)..s,\ r(j)..t-1) & +\ \gamma(\varepsilon \to D[t]), \\ \mathrm{f_d}(r(i)..s-1,\ r(j)..t-1) & +\ \gamma(P[s] \to D[t]), \\ \min_{t_k}\{\mathrm{t_d}(s,t_k)\} & 1 \le k \le n_t, \\ \min_{t_k}\{\mathrm{s_f_d}(r(i)..s-1, r(j)..t_k)\} & 1 \le k \le n_t \end{array} \right\}$$

where $D[t_k]$, $1 \leq k \leq n_t$, are children of $D[t]$. If $D[t]$ is a leaf, that is $t = r(j)$, then only the first three expressions are present. We define the *suffix forest distance* between F_P and F_D, forests in the pattern P and the data tree D respectively, denoted $\text{s_f_d}(F_P, F_D)$, as the distance between F_P and $\bar{F}_D$, where $\bar{F}_D$ is a sub-forest of F_D with some consecutive complete subtrees removed from the left all having the same parent. Formally we have that

$$\text{s_f_d}(F_P, F_D) = \min_{\bar{F}_D} \{\text{f_d}(F_P, \bar{F}_D)\}$$

From a computational point of view, it can be proved that

$$\text{s_f_d}(r(i)..s, r(j)..t) = \begin{cases} \min \left\{ \begin{array}{l} \text{f_d}(r(i)..s,\ \phi), \\ \text{f_d}(r(i)..s,\ r(j)..t) \end{array} \right\} \\ \textbf{iff } r(t) = r(j) \\ \\ \min \left\{ \begin{array}{ll} \text{s_f_d}(r(i)..s-1,\ r(j)..t) & + \gamma(P[s] \to \varepsilon), \\ \text{s_f_d}(r(i)..s,\ r(j)..t-1) & + \gamma(\varepsilon \to D[\hat{t}]), \\ \text{s_f_d}(r(i)..r(s)-1,\ r(j)..r(t)-1) & + \text{t_d}(s,t) \end{array} \right\} \\ \textbf{otherwise} \end{cases}$$

To compute $t_d(P, D)$ it will be sufficient to take into account that

$$\text{t_d}(P, D) = \text{f_d}(\text{root}(P)..r(\text{root}(P)), \text{root}(D)..r(\text{root}(D)))$$

Time bound is $\mathcal{O}(|P| \times |D| \times \min(depth(P), leaves(P)) \times \min(depth(D), leaves(D)))$ in the worst case, where $|P|$ (resp. $|D|$) is the number of nodes in the pattern tree P (resp. in the data tree D).

4 Approximate VLDC matching in shared forest

To start with, let P be a labeled ordered tree where some structural details has been omitted, and D an AND-OR graph, both of them built using our parsing frame. We shall identify P with a query and D with a part of the syntactic representation for a textual database with a certain degree of ambiguity. The presence of OR nodes in D has two main implications in our work: Firstly, there will exist situations where we must handle simultaneous values for some forest distances and, secondly, the parser may share some structures among the descendants of the different branches in an OR node. We shall now present the manner we calculate the distance between a pattern tree and the set of trees that are represented within the AND-OR graph, and how to take advantage of the shared structures created by the parser.

Let $P[s]$ be the current node in the inverse postorder for P, and $i \in \text{anc}(s)$ a r_keyroot. Given an OR node $D[k]$ we can distinguish two situations, depending on the situation of this OR node and the situation of the r_keyroots of D.

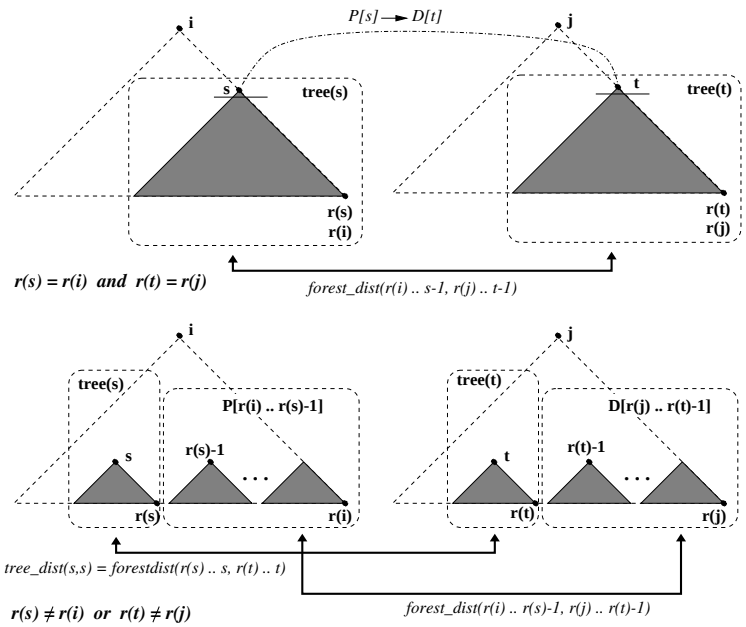


Fig. 4. The forest distance in our proposal

4.1 Sharing into a same r_keyroot

Let $D[t']$ and $D[t'']$ be the nodes we are dealing with in parallel for two branches labeled $D[k']$ and $D[k'']$ of the OR node $r(D[k])$. We have that $j \in anc(t') \cap anc(t'')$, that is, the tree rooted at the r_keyroot $D[j]$ includes the OR alternatives $D[k']$ and $D[k'']$.

Such a situation is shown in Fig. 5 using a classic representation and the AND-OR graphs. Here, the lightly shaded part refers to nodes whose distance have been computed in the inverse postorder before the OR node $D[k]$. The heavily shaded part represents a shared structure. The notation "• • •" in figures representing AND-OR graphs, expresses the fact that we descend along the rightmost branch of the corresponding tree.

We shall assume that nodes $D[r(t') - 1]$ and $D[r(t'') - 1]$ are the same, that is, their corresponding subtrees are shared. So, $D[r(t')]$ (resp. $D[r(t'')]$) is the following node in $D[k']$ (resp. $D[k'']$) to deal with once the distance for the shared structure has been computed.

At this point, our aim is to compute the value for $f_d(r(i)..s,\ r(j)..\hat{t}),\ \hat{t} \in \{t',\ t''\}$, proving that we can translate parse sharing on sharing on computations for these distances.

Since we have assumed there is a shared structure between $D[r(\hat{t})]$ and $D[r(j)]$, we conclude that $r(j) \neq r(\hat{t})$ and the values for $f_d(r(i)..s,\ r(j)..\hat{t}),\ \hat{t} \in \{t',\ t''\}$ are given by:

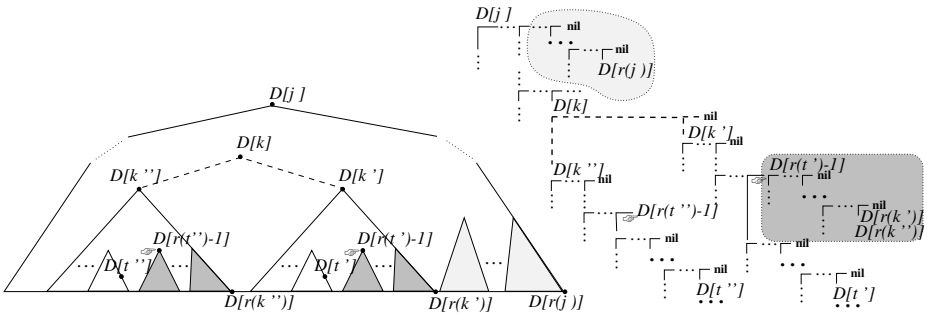


Fig. 5. Sharing into a same r_keyroot

$$\text{f_d}(r(i)..s, r(j)..\hat{t}) = \min \left\{ \begin{array}{ll} \text{f_d}(r(i)..s-1,\ r(j)..\hat{t}) & +\ \gamma(P[s] \to \varepsilon), \\ \text{f_d}(r(i)..s,\ r(j)..\hat{t}-1) & +\ \gamma(\varepsilon \to D[\hat{t}]), \\ \text{f_d}(r(i)..r(s)-1,\ r(j)..r(\hat{t})-1) & +\ \text{t_d}(s, \hat{t}) \end{array} \right\}$$

where $\hat{t} \in \{t',\ t''\}$. We can interpret this three alternatives as follows:

1. The values for $f_d(r(i)..s-1,\ r(j)..\hat{t}),\ \hat{t} \in \{t',\ t''\}$ have been computed by the approximate matching algorithm in a previous step. So, in this case, parse sharing has not consequences on the natural computation for the distances.
2. Two cases are possible in relation to the nature of nodes $D[\hat{t}],\ \hat{t} \in \{t',\ t''\}$, these are:
   - If both nodes are leaves, then $r(\hat{t}) = \hat{t}$. As a consequence, we also have that
   $$D[t'-1] = D[r(t')-1] = D[r(t'')-1] = D[t''-1]$$
   and the values $f_d(r(i)..s,\ r(j)..\hat{t}-1),\ \hat{t} \in \{t',\ t''\}$ are also the same.
   - Otherwise, following the inverse postorder, we would arrive at the rightmost leaves of $D[t']$ and $D[t'']$, where we could apply the reasoning considered in the previous case.
3. Values for the distances $f_d(r(s)..r(i)-1,\ r(j)..r(\hat{t})-1),\ \hat{t} \in \{t',\ t''\}$ are identical, given that nodes $D[r(\hat{t})-1],\ \hat{t} \in \{t',\ t''\}$ are shared by the parser.

A similar reasoning can be applied to compute the values for s_f_d, avoiding redundant computations.

4.2 Sharing between different r_keyroots

We have that $j' \in anc(t')$ and $j'' \in anc(t'')$, with $j' \neq j''$, are two r_keyroots. We also have an OR node $D[k]$ being a common ancestor of these two nodes. We suppose that the r_keyroots are in different branches, that is, there exists a r_keyroot, $D[j']$ (resp. $D[j'']$), in the branch labeled $D[k']$ (resp. $D[k'']$).

Our aim now is to compute the value for distances $f_d(r(i)..s,\ r(\hat{\jmath})..\hat{t})$, where pairs $(\hat{\jmath}, \hat{t})$ are in $\{(j',\ t'),\ (j'', t''\ )\}$. Formally, we have that these values are given by:

$$
\mathrm{f_d}(r(i)..s, r(\hat{\jmath})..\hat{t}) = \begin{cases}
\begin{cases}
\min \left\{ \begin{array}{ll}
\mathrm{f_d}(r(i)..s-1,\ r(\hat{\jmath})..\hat{t}) & +\ \gamma(P[s] \to \varepsilon), \\
\mathrm{f_d}(r(i)..s,\ r(\hat{\jmath})..\hat{t}-1) & +\ \gamma(\varepsilon \to D[\hat{t}]), \\
\mathrm{f_d}(r(i)..s-1,\ r(\hat{\jmath})..\hat{t}-1) & +\ \gamma(P[s] \to D[\hat{t}]), \\
\mathrm{f_d}(\phi, D[r(\hat{\jmath})]..\hat{t}-1) & +\ \min_{\hat{t}_k}\{\mathrm{t_d}(s, \hat{t}_k) - \\
 & \quad \mathrm{t_d}(\phi, \hat{t}_k)\} \quad 1 \le k \le n_{\hat{t}}
\end{array} \right\} \\
\textbf{iff } P[s] = | \\
\\
\min \left\{ \begin{array}{ll}
\mathrm{f_d}(r(i)..s-1,\ r(\hat{\jmath})..\hat{t}) & +\ \gamma(P[s] \to \varepsilon), \\
\mathrm{f_d}(r(i)..s,\ r(\hat{\jmath})..\hat{t}-1) & +\ \gamma(\varepsilon \to D[\hat{t}]), \\
\mathrm{f_d}(r(i)..s-1,\ r(\hat{\jmath})..\hat{t}-1) & +\ \gamma(P[s] \to D[\hat{t}]), \\
\min_{\hat{t}_k}\{\mathrm{t_d}(s, \hat{t}_k), & \quad 1 \le k \le n_{\hat{t}}, \\
\min_{\hat{t}_k}\{\mathrm{s_f_d}(r(i)..s-1, r(\hat{\jmath})..\hat{t}_k), & \quad 1 \le k \le n_{\hat{t}}
\end{array} \right\} \\
\textbf{iff } P[s] = \wedge \\
\\
\min \left\{ \begin{array}{ll}
\mathrm{f_d}(r(i)..s-1,\ r(\hat{\jmath})..\hat{t}) & +\ \gamma(P[s] \to \varepsilon), \\
\mathrm{f_d}(r(i)..s,\ r(\hat{\jmath})..\hat{t}-1) & +\ \gamma(\varepsilon \to D[\hat{t}]), \\
\mathrm{f_d}(r(i)..s-1,\ r(\hat{\jmath})..\hat{t}-1) & +\ \gamma(P[s] \to D[\hat{t}])
\end{array} \right\} \\
\textbf{otherwise}
\end{cases} \\
\textbf{iff } r(s) = r(i) \textbf{ and } r(\hat{t}) = r(\hat{\jmath}) \\
\\
\min \left\{ \begin{array}{ll}
\mathrm{f_d}(r(i)..s-1,\ r(\hat{\jmath})..\hat{t}) & +\ \gamma(P[s] \to \varepsilon), \\
\mathrm{f_d}(r(i)..s,\ r(\hat{\jmath})..\hat{t}-1) & +\ \gamma(\varepsilon \to D[\hat{t}]), \\
\mathrm{f_d}(r(i)..r(s)-1,\ r(\hat{\jmath})..r(\hat{t})-1) & +\ \mathrm{t_d}(s, \hat{t})
\end{array} \right\} \\
\textbf{otherwise}
\end{cases}
$$

The situation, shown in Fig. 6, makes possible $r(s) = r(i)$ and $r(\hat{t}) = r(\hat{\jmath})$. In this first case, we can assume that a tail of sons is shared by nodes $D[\hat{t}]$, $\hat{t} \in \{t',\ t''\}$. We can also assume that this tail is proper given that, otherwise, our parser guarantees that the nodes $D[\hat{t}]$, $\hat{t} \in \{t',\ t''\}$ are also shared.

Taking into account our parsing strategy, which identifies syntactic structures and computations, we conclude that the distances $f_d(r(i)..s,\ r(\hat{\jmath})..\hat{t})$, with $(\hat{\jmath}, \hat{t}) \in \{(j',\ t'),\ (j'', t''\ )\}$ do not depend on previous computations over the shared tail, as is shown in the left-hand-side of Fig. 6. So, this sharing has no consequences on the calculus, although it will have effects on the computation of distances for nodes in the rightmost branch of the tree immediately to the left of the shared tail of sons, which is denoted by a double pointed line in Fig. 6, as we shall shown immediately.

We consider now the second case, that is, the computation of the forest distance when $r(\hat{t}) \neq r(\hat{\jmath})$, such as is shown in Fig. 7. Here, in relation to each one of the three alternative values to compute the minimum, we have that:

1. The values for $f_d(r(i)..s-1,\ r(\hat{\jmath})..\hat{t})$, $(\hat{\jmath}, \hat{t}) \in \{(j',\ t'),\ (j'', t''\ )\}$ have been computed by the approximate matching algorithm in a previous step and parse sharing does not affect the computation for distances.

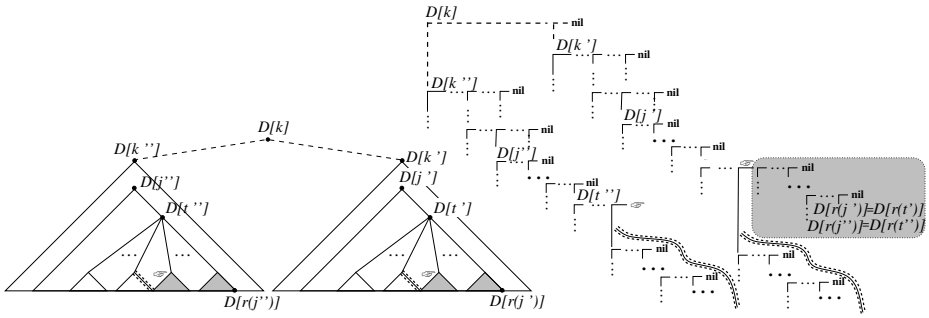


Fig. 6. Sharing between different r_keyroots (first case)

2. We distinguish two cases in relation to the nature of nodes $D[\hat{t}]$, $\hat{t} \in \{(t',\ t'')$. We shall apply the same reasoning considered when we had an only r_keyroot:
   - If both nodes are leaves, then $r(\hat{t}) = \hat{t}$. As a consequence, we also have that

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

     and therefore the values for distances $f_d(r(i)..s,\ r(\hat{\jmath})..\hat{t}-1)$ with $(\hat{\jmath}, \hat{t}), \in \{(j',\ t'),\ (j'', t''\ )\}$, are also the same.
   - Otherwise, following the inverse postorder, we arrive to the rightmost leaves of $D[t']$ and $D[t'']$, where we can apply the reasoning considered in the previous case.

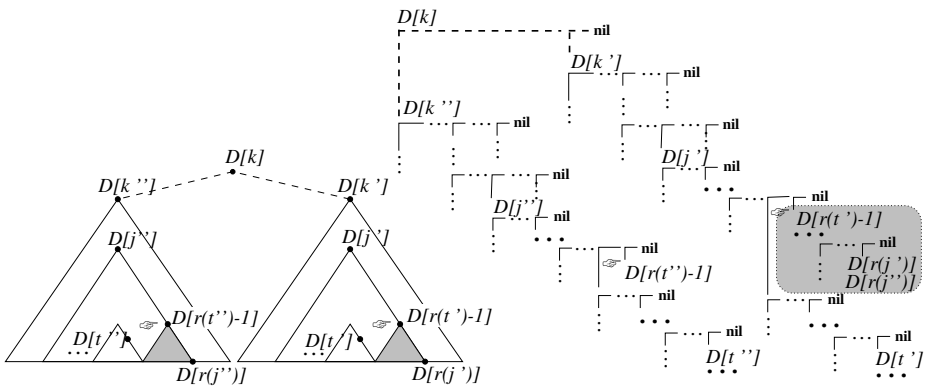


Fig. 7. Sharing between different r_keyroots (second case)

3. Values for the distances $f_d(r(i)..r(s)-1,\ r(\hat{t})..r(\hat{t})-1)$, $\hat{t} \in \{t',\ t''\}$ are identical, given that the trees rooted by nodes $D[r(\hat{t})-1]$, $\hat{t} \in \{t',\ t''\}$ are shared by the parser.

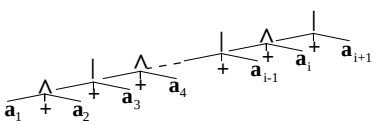
As in the case of sharing on a same r_keyroot, a similar reasoning can be applied to compute the values for s_f_d, avoiding redundant computations.

5 Experimental results

We take a simple example to illustrate our discussion: the language of arithmetical expressions. We compare our proposal with Zhang *et al.* [10], considering two deterministic grammars, $\mathcal{G}_L$ and $\mathcal{G}_R$, representing respectively the left and right associative versions for the arithmetic operators; and one non-deterministic one $\mathcal{G}_N$. To simplify the explanation, we focus on matching phenomena assuming that parsers are built using ICE [8]. Lexical information is common in all cases, and tests have been applied on target inputs of the form $a_1 + a_2 + \ldots + a_i + a_{i+1}$, with i even, representing the number of addition operators. These programs have a number of ambiguous parses which grows exponentially with i. This number is:

$$C_0 = C_1 = 1 \quad \text{and} \quad C_i = \binom{2i}{i} \frac{1}{i+1}, \text{ if } i > 1$$

As pattern, we have used deterministic parse trees of the form



In the deterministic case, patterns are built from the left-associative (resp. right-associative) interpretation for $\mathcal{G}_L$ (resp. $\mathcal{G}_R$), which allows us to evaluate the impact of traversal orientation in the performance. So, the rightmost diagram in Fig. 8 proves the adaptation of our proposal (resp. Zhang *et al.* algorithm) to left-recursive (resp. right-recursive) derivations, which corroborates our conclusions.

In the non-deterministic case, patterns are built from the left-associative interpretation of the query, which is not relevant given that rules in $\mathcal{G}_N$ are symmetrical. Here, we evaluate the gain in efficiency due to sharing of computations in a dynamic frame, such as is shown in the leftmost diagram of Fig. 8.

6 Conclusions

Approximate tree matching can be adapted to deal with shared data forest and incomplete pattern trees, to give rise to approximate VLDC pattern matching. In practice, this approach can reduce the cost of evaluating queries in sophisticated retrieval systems, where retrieval functions based on classic pattern matching cannot reach optimal results because it is impossible to estimate the exact representation of documents or requests and additional simplifying assumptions are necessary.

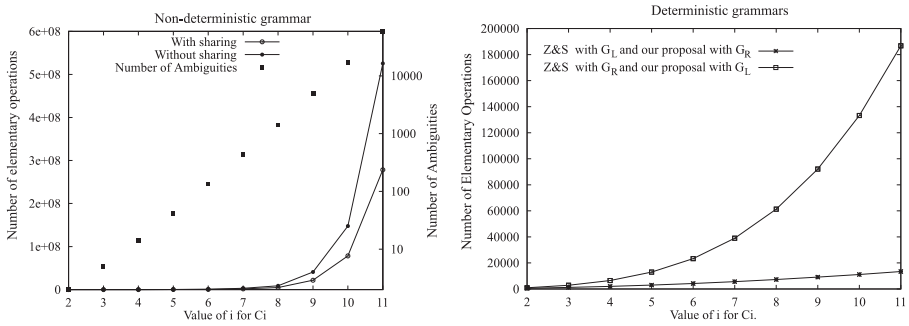


Fig. 8. Results on approximate VLDC matching

References

1. Billot, S., Lang, B.: The structure of shared forest in ambiguous parsing. *Proc. of* 27th *Annual Meeting of the ACL*, 1989.
2. Kilpelainen, P., Mannila, H.: Grammatical Tree Matching. *Lecture Notes in Computer Science* **644** (1992) 159–171.
3. Kilpelainen, P.: Tree Matching Problems with Applications to Structured Text Databases. *Ph.D. Thesis*, Department of Computer Science, University of Helsinki, 1992. Helsinki, Finland
4. Smeaton, Alan F. : Incorporating Syntactic Information into a Document Retrieval Strategy: An Investigation. *Proc. the* 9th *Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 103–113, 1986.
5. Smeaton, A.F. and van Rijsbergen, C.J.: Experiments on Incorporating Syntactic Processing of User Queries into a Document Retrieval Strategy. *Proc. of the* 11th *International ACM-SIGIR Conference on Research and Development in Information Retrieval*, pp. 31-54. Grenoble, France, 1988.
6. Smeaton, A.F, O'Donell, R. and Kelley,F.: Indexing Structures Derived from Syntax in TREC-3: System Description. *Proc. of* 3rd *Text REtrieval Conference* (TREC-3), D.K. Harman (ed.), NIST Special Publication, 1994.
7. Vilares, M., Cabrero, D., Ribadas F.J.: Approximate matching in shared forest. *Proc. of Sixth International Workshop on Natural Language Understanding and Logic Programming* pp. 59-72, Las Cruces, NM (USA), 1999.
8. Vilares, M., Dion, B.A.: Efficient incremental parsing for context-free languages. *Proc. of the* 5th *IEEE International Conference on Computer Languages* (1994) 241–252, Toulouse, France.
9. Zhang, K., Shasha, D.: Simple fast algorithms for the editing distance between trees and related problems. *SIAM Journal on Computing* (1989) **18** 1245–1262.
10. Zhang, K. and Shasha, D. and Wang, J.T.L. Approximate Tree Matching in the Presence of Variable Length Don't Cares. *Journal of Algorithms*, pages 33–66, vol **16 (1)**, 1994