Layout based document image retrieval by means of XY tree reduction

Simone Marinai Emanuele Marino Giovanni Soda
DSI - University of Florence (Italy)
E-mail: marinai@dsi.unifi.it

Abstract

We analyze a system for the retrieval of document images on the basis of layout similarity. Layout objects are extracted and represented with the XY tree. Page similarity is computed with a tree-edit distance algorithm. The peculiarity of the approach is the use of tree grammars to model the variations in the tree which are due to segmentation algorithms or to structural differences between documents with similar layout. A few class-independent grammatical rules are used to modify each tree and obtain a reduced tree that is supposed to preserve the most relevant features of the page.

1. Introduction

Libraries and archives digitized large collections of historical documents to preserve their holdings by reducing the need for manual access to such valuable items. Less emphasis has been dedicated to the retrieval of relevant information from digital collections. Document Image Retrieval (DIR) aims at finding relevant documents relying on image features only. Important sub-tasks are the retrieval by layout similarity and the text based retrieval. Few methods contemplated the retrieval by layout similarity. In most cases a fixed-size feature vector is obtained by computing some features in the regions defined by a grid superimposed to the page [1, 2]. To overcome the problems due to the choice of a fixed grid size, hierarchical representations of the page layout, like the XY tree, have been considered [3, 4]. One assumption in XY tree based DIR is the possibility of retrieving similar pages considering some tree similarity. In [4], the tree edit distance is considered for the retrieval of forms represented by means of XY trees. The tests performed are made on a small data-set. When dealing with less constrained documents (e.g. journal pages) there are few invariant parts and trees belonging to the same class are likely to have a larger distance. This is due to both the presence of noise in the images and to actual differences in page layouts.

In this paper we propose one solution to overcome the variability in XY tree representation that relies on the use of tree-grammar based transformations of the XY trees. In [5] we proposed the use of tree-grammars for page classification. One important limit of this approach for the use in DIR is the presence of class-dependent rules. The grammar proposed in this paper is designed to take into account typical errors and differences in page layout which are reflected in the corresponding XY trees. Two approaches for the use of tree-grammars in DIR are described. In both approaches in the indexing phase the XY tree is computed from each page and processed according to the tree-grammar. In the retrieval phase the user provides to the system a prototype document (with a query by example mechanism) and a XY tree is computed for the query page and processed by means of the tree-grammar. A similarity measure is then computed between the query page and the pages previously indexed.

The first approach is based on the idea of query expansion: given one query page provided by the user we apply typical transformations to the query XY tree to simulate variations in the tree that might correspond to actual variations in document images. Documents in the dataset are afterwards ranked on the basis of the similarity with the whole set of trees obtained from the query one.

The second approach takes into account a different grammar to reduce the complexity of trees by removing tree-structures that usually bring few information. The similarity is computed by evaluating the distance between the reduced query tree and reduced trees in the database. In [6] we compared the two approaches on a single-book dataset, with a simpler approach to tree reduction. In this paper we propose the use of a general grammar, and test it on an broad dataset containing a XIXth Century book and technical journals from the UW-III dataset.

The similarity among trees is evaluated by means of the tree edit distance, that is based on the evaluation of the number of edit operations needed to transform one tree into another. The distance between two trees is defined as the cost of the minimum-cost set of operations that are required to transform one tree into the other. Zhang [7] proposed an algorithm to compute the tree edit distance that we use to measure the effectiveness of the grammar based tree transformation for DIR.

The paper is organized as follows: in Section 2 we describe the syntax of the tree grammar, in Section 3 the algorithms proposed to modify the trees by means of the grammar are analyzed. Section 4 summarizes the rules considered in the experimentas reported in Section 5. A final discussion is drawn in Section 6.

2. XY tree grammar

Tree grammars [8] are similar to string grammars except that the basic objects are trees instead of strings. More formally, a **tree grammar** $G = (\mathcal{S}, \mathcal{N}, \mathcal{T}, \mathcal{P})$ is defined by a starting symbol $\mathcal{S}$ ($\mathcal{S} \in \mathcal{N}$), a set $\mathcal{N}$ of nonterminal symbols, a set $\mathcal{T}$ of terminal symbols, a set $\mathcal{P}$ of production rules of the form $\alpha \rightarrow \beta$ where α contains at least one nonterminal. Similarly to the XSLT terminology[1] we refer to α as a *pattern* and to β as a *template*. Generally speaking the pattern defines the trees where the rule can be applied and the template describes how to build the output tree. Each labeled tree is represented as a string by using a pre-fix notation. A label describes a region, like the type of XY-cut for internal nodes and the region content for leaves. The labels can encode additional information such as the region size. The rules can contain user-defined *logical predicates* and *alteration functions*. A *logical predicate* (Table 1) is part of the rule pattern, it checks a local property of one tree (related to arrangement, number, and label of nodes) and returns a boolean value. An *alteration function* is part of the rule template and modifies the tree structure. For instance the function $Opt(T)$ randomly adds a T leaf on a given position. The symbol $\#$ corresponds to the empty tree. Rules have the form:

$$\texttt{node\_name} : pattern \rightarrow \texttt{node\_name} : template \quad (1)$$

where $pattern$ and $template$ contain a tree (with predicates and alteration functions) defined with a pre-fix notation. For instance in the following rule one hs node with two children (one T and one L) is replaced with a T leaf:

$$n_1 : hs[n_2 : (T), n_3 : (L)] \rightarrow n_1 : T[n_2 : \#, n_3 : \#] \quad (2)$$

3. Tree manipulation

Two approaches can be considered when using a tree grammar to modify a XY tree. Either the query tree can be "expanded" or every tree in the database, in addition to the query one, can be reduced.

3.1. Query expansion

After defining an appropriate set of rules, it is possible to modify one tree so as to reflect actual differences in trees built from similar pages. To avoid an excessive distortion in generated trees we assign a cost to the deformation made by each rule and we allow a maximum accumulated cost for each generated tree.

1 See: "http://www.w3.org/TR/xslt"

Predicate	Meaning
`Lup` $value$	Node level greater than $value$
`AP` $value$	Region with area lower than $value$ % of the whole image
`FimT`	The children are all leaves and at least one leaf is T
`FimI`	The children are all leaves and at least one leaf is I

Table 1. Some logical predicates of interest for the rules discussed in the paper.

Algorithm 1 describes the function `ExpandQ`(t_Q) that performs query expansion of the query tree t_Q and produces a set of trees (Exp) obtained by applying the grammar described in Rule_Set (the grammar is reported in Section 4). Before calling `ExpandQ`, Exp is initialized with the empty set and Cost(t_Q) is set to 0. In the algorithm, a given rule r with pattern that matches one tree t will be applied (giving rise to tree T) when the accumulated cost ($Cost$) is lower than the maximum allowed cost ($MaxCost$). T is a list of trees that is obtained as the result of the possible applications of r to t. T is produced by `Apply`(t, r) that also updates the value of Cost(T). `Apply` returns a null value when the conditions in the pattern do not hold.

Algorithm 1 ExpandQ(t)
  $Exp \leftarrow Exp \cup t$
  foreach $r \in$ Rule_Set **do**
    $T \leftarrow$ Apply(t,r)
    if ($T \neq \emptyset$ AND Cost(T) $<$ MaxCost)
      foreach $\bar{t} \in T$ **then** ExpandQ($\bar{t}$)
    else Return

An expanded tree ($\bar{t}$ in the algorithm) can be expanded again. However, the association of a cost to each rule avoids the presence of infinite recursions and bounds the maximum number of trees that may be generated. After the query expansion the set Exp can be used for computing the similarity between each tree in the database and t_Q.

3.2. Tree reduction

Tree reduction is based on the assumption that similar trees share a common structure. With appropriate rules (see Section 4) the query tree can be simplified so as to obtain a smaller tree that replaces the original one. One example of a typical rule is the substitution of a set of leaves labeled as text with a unique leaf labeled as text as well. In general, several rules can be applied to one tree and several alternatives are possible. As a consequence, a hierarchy of trees can be obtained starting from a given t by means of Algorithm 1. When the algorithm terminates we may select the

Class name	Description	DB
Biblio	Bibliography page	W
Image	One image in the page	G W
ImageText2	One image above a two columns text	G W
Issue2	First page of a chapter	G
SecE2	End-of-section page	G
SecM2	Title of a new paragraph	G
TableText2	Table and text on 2 columns	W
Text2	Text on 2 columns (no images)	G W
Text2Image	Image and text on 2 columns	G W
Text2Table	One table over a two columns text	W
Title	First page of the article	W

Table 2. Classes of pages in the dataset. The DB column points out the databases where the class is present (G = "Gallica", W= UWIII).

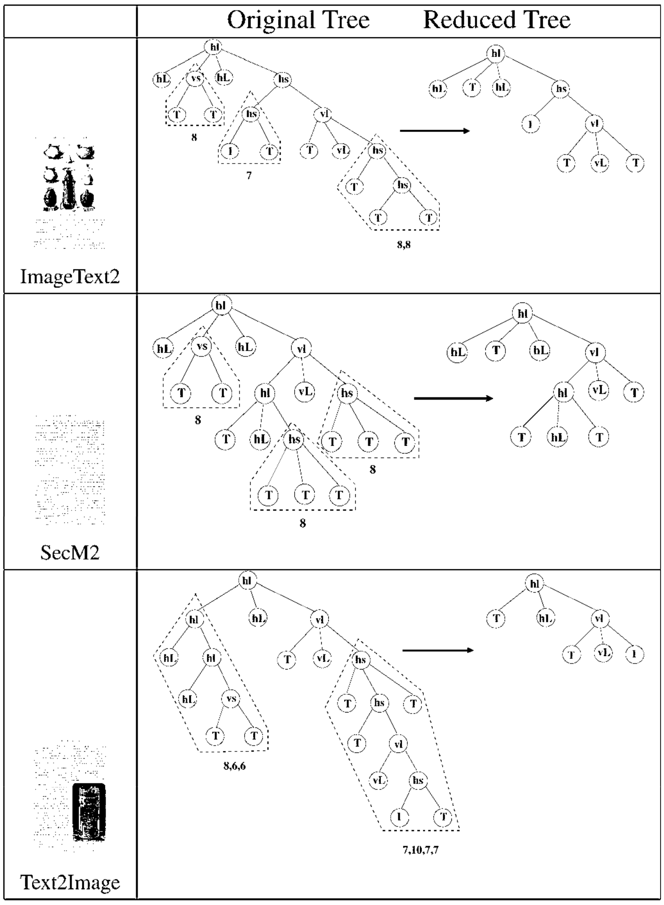

Figure 1. Documents of three classes for the Gallica dataset. We point out the subtrees modified by the grammar (with the rules applied), and the reduced trees.

most appropriate tree by looking, for instance, to the size of all the generated trees. One significant difference with respect to Section 3.1 is that the rules are usually contractive ones (the number of nodes in the template is lower or equal to the number of nodes in the pattern), therefore, even by setting $CostMax$ to a large value the algorithm terminates. However, it frequently occurs that a large number of trees belong to the hierarchy and it may be not appropriate to generate every possible tree. This computational problem is more critical with respect to the expansion case since now we must process all the trees in the database and not only the query tree.

To address these efficiency issues we devised an approach that regards the generation of reduced trees as a search algorithm and implicitly interprets the hierarchy of trees as a search space. To select the most promising rules a simple heuristic function has been defined (Eq. 3) which assigns a value to each tree considering the number of nodes at each level (m is the height of the tree and n_k is the number of nodes at the level k).

$$h(t) = 1 + \sum_{k=0}^{m} \sum_{i=1}^{n_k} k \cdot i \tag{3}$$

Only the rule with the lowest value for h is applied. In so doing to reduce a generic tree t a Best-First search is applied as summarized in Algorithm 2. At the end of the algorithm one reduced tree, RT, is returned. The algorithm terminates when there are no rules to apply at any point of the current tree. The computation of reduced trees is a computationally intensive step that is applied once for each dataset during its indexing. The similarity is computed by comparing the simplified query tree and the simplified trees in the database.

Algorithm 2 ReduceTree(t)
$\bar{t} \leftarrow \emptyset$; $w \leftarrow \infty$; $RT \leftarrow t$
foreach $r \in$ Rule_Set **do**
  $T \leftarrow$ Apply(t,r)
  if ($T \neq \emptyset$)
  foreach $\hat{t} \in T$ **do**
    if ($h(\hat{t}) < w$)
      $\bar{t} \leftarrow \hat{t}$; $w \leftarrow h(\bar{t})$
if ($\bar{t} \neq \emptyset$) ReduceTree($\bar{t}$)

4. Rule sets

When dealing with the two methods described in Section 2 appropriate rule sets need to be defined. The meaning of labels is: T for text region, I for image region, hL (vL) for horizontal (vertical) ruling line. Concerning internal nodes we have: hs (vs) for one cut along horizontal (vertical) white space; hl (vl) for one cut along horizontal (vertical) line; l denotes either hl or vl; similarly s denotes either hs or vs.

The set of rules devised for **query expansion** is:

1. $n_1 : T(Lup\,1) \rightarrow n_1 : hs[T, T, Opt(T)]$
   Replace a T leaf with a subtree having a hs root and 2 or 3 leaves labeled as T; the number of leaves is randomly selected.
2. $n_1 : T(Lup\,1) \rightarrow n_1 : hl(T, hL)$
   Replace a T leaf with a subtree having a hl root and 2 leaves labeled as T and L, respectively.
3. $n_1 : T(Lup\,1) \rightarrow n_1 : hl(hL, T)$
   Replace a T leaf with a subtree having a hl root and 2 leaves labeled as L and T, respectively.

The set of rules devised for **tree reduction** is:

1. $n_1 : I(AP\,1) \rightarrow n_1 : T$
   A small I region is replaced with a T region.
2. $n_1 : T(AP\,1) \rightarrow n_1 : \#$
   A small T region is removed.
3. $n_1 : L(AP\,1) \rightarrow n_1 : \#$
   A short line is removed.
4. $n_1 : s[n_2 : (L)+] \rightarrow n_1 : hL[n_2 : \#]$
   A node (hs or vs) with only L children is replaced with a L leaf (all the children are deleted).
5. $n_1 : l[n_2 : (T), n_3 : (L)] \rightarrow n_1 : T[n_2 : \#, n_3 : \#]$,
6. $n_1 : l[n_2 : (L), n_3 : (T)] \rightarrow n_1 : T[n_2 : \#, n_3 : \#]$
   One l node with two children (one L and one T) is replaced with a T leaf.
7. $n_1 : s(FimI) \rightarrow n_1 : I$,
8. $n_1 : s(FimT) \rightarrow n_1 : T$
   When one children of s is I (T), then s becomes a I (T) leaf.
9. $n_1 : l[n_2 : I, n_3 : L] \rightarrow n_1 : I[n_2 : \#, n_3 : \#]$,
10. $n_1 : l[n_2 : L, n_3 : I] \rightarrow n_1 : I[n_2 : \#, n_3 : \#]$
    One l node with two children (one L and one I) is replaced with a I leaf.
11. $n_1 : s(isLeaf) \rightarrow n_1 : \#$
    A node (hs or vs) is removed if it is a leaf.

5. Experimental results

We considered two datasets for the experiments. The "Gallica" dataset contains the 601 pages of a XIXth Century book [2] that is part of a multi-volume encyclopedia. The second dataset contains all the 311 pages from the W0* series of the UWIII database (B/W images of English Journals). The pages of the datasets can be grouped in the classes described in Table 2. Two experiments have been performed with the aim of evaluating the Precision & Recall plots when making queries with pages of user-defined classes. On both experiments for each class we grouped 50 randomly selected pages. For each page in the groups we made a query and we computed the precision-recall plots for the methods that we compared: the standard method (without the use of the grammar), the query expansion (only in the first experiment), and the tree reduction. Plots reported on this paper are computed by averaging the 50 plots obtained for each query. The retrieval has been performed by considering the whole set of pages in each dataset, and not only the classes considered for the query.

In the first experiment we considered the "Gallica" dataset and compared the approaches described in Section 3 using the rules listed in Section 4. For the tree reduction we did not use rules 2, 3, and 11, that are needed for noisy trees. We focused on three types of pages that are very common: *ImageText2*, *Text2Image*, and *SecM2* (see Figure 1 for examples of pages). Additional results of this experiment are reported in [6]. The results achieved with query expansion are similar to the baseline. For $SecM2$ most errors are due to $Text2$ pages erroneously evaluated similar to $SecM2$ ones. This is probably due to the large size of expanded trees that does not allow to easily differentiate small differences with the tree-edit distance. Encouraging results can be remarked by looking at the reduction approach. In this case we always have a clear improvement of performance. We therefore made the next experiment comparing only the tree-reduction approach with the baseline.

In the second experiment we refined the reduction rules and added rules 2, 3 and 11 to cope with small regions mainly due to segmentation errors. In the first row of Figure 2 we considered again the "Gallica" data-set with the new rules and we still achieved good results. Comparing the two rows of Figure 2 we can observe that the results on the UW dataset are significantly worse than those in the Gallica data. This is not surprising, considering that the latter documents are more uniform than the first ones. In both cases the reduction rules deal in a good way with most distortions due to noise. However, the style variations in the UW dataset are harder to model. In Figure 2 we shown the results achieved for a few classes. With other classes we obtained similar results, with the exception of the $Text2$ class. For this class we observed worst results when using reduction rules with respect to the baseline method. This is probably due to excessive reduction of some trees that are considered as $Text2$. Even if the $Text2$ class is the most frequent in the dataset we feel that a typical user would be less interested in retrieving pages with this standard layout and containing only text. In this case a text-based retrieval would be

2 Downloaded from the web site of the *National Library of France*.

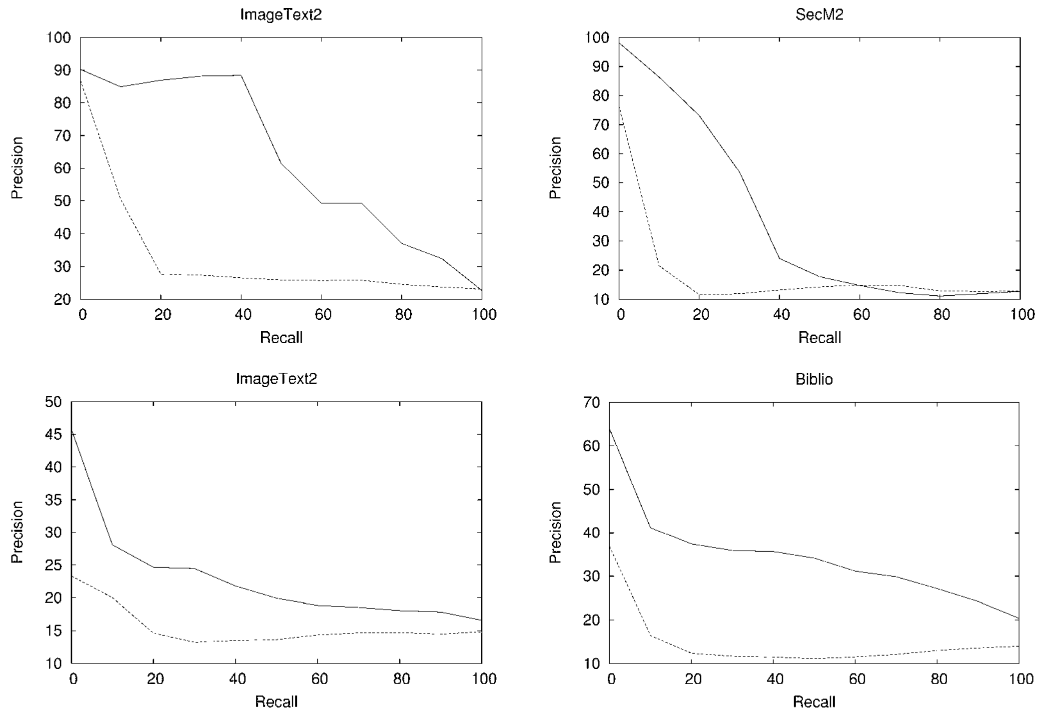

Figure 2. Comparison of Precision & Recall plots for the "Gallica" dataset (first row) and UW dataset (second row). Continuous line: tree reduction; dotted line: the grammar is not used.

more appropriate to answer the user's queries.

6. Conclusions

In this paper we deal with layout based retrieval of document images belonging to digital libraries. The method relies on a tree-based representation of the page layout and computes page similarity on the basis of tree-edit distance. To improve the similarity computation a tree grammar is introduced so as to model the typical distortions that occur in actual documents. A reduction algorithm and general rules have been devised and tested on a mixed dataset containing a book of the XIXth Century and technical journals from the UWIII database. The proposed rules work quite well on such heterogeneous dataset providing an improvement in the retrieval performance on most cases.

The current research includes the study of algorithms for the semi-automatic learning of most appropriate grammatical rules. Moreover, we plan to replace the tree edit distance algorithm currently in use with more efficient algorithms that allow to perform approximate nearest neighbor search between trees [9].

References

[1] J.Hu, R.Kashi, and G.Wilfong, "Comparision and classification of documents based on layout similarity," *Information Retrieval*, vol. 2, pp. 227–243, May 2000.

[2] A. Tzacheva, Y. El-Sonbaty, and E. A. El-Kwae, "Document image matching using a maximal grid approach," in *Proceedings of the SPIE Document Recognition and Retrieval IX*, pp. 121–128, 2002.

[3] F. Cesarini, S. Marinai, and G. Soda, "Retrieval by layout similarity of documents represented with MXY trees," in *Document Analysis Systems V*, pp. 353–364, Springer Verlag, LNCS 2423, 2002.

[4] P. Duygulu and V. Atalay, "A hierarchical representation of form documents for identification and retrieval," *IJDAR*, vol. 5, pp. 17–27, November 2002.

[5] S. Baldi, S. Marinai, and G. Soda, "Using tree grammars for training set expansion in page classification," in *Proc. 7th ICDAR*, pp. 829–833, 2003.

[6] S. Marinai, E. Marino, and G. Soda, "Layout based document image retrieval in digital libraries," in *7th Int. Workshop Audio-Visual Content and Information Visualization in Digital Libraries (AVIVDiLib '05)*, pp. 67–76, 2005.

[7] K. Zhang and D. Shasha, "Simple fast algorithms for the editing distance between trees and related problems," *SIAM J. Computing*, vol. 18, pp. 1245–1262, December 1989.

[8] R. C. Gonzalez and M. C. Thomason, *Syntactic Pattern Recognition - an introduction*. Addison-Wesley, 1978.

[9] D. Shasha, J. Wang, H. Shan, and K. Zhang, "ATreeGrep: Approximate searching in unordered trees," in *14th Int. Conf. on Scientific and Statistical Database Management*, pp. 89–98, 2002.