Highlighting latent structure in documents

H. Folch, B. Habert, M. Jardino, N. Pernelle, M.C. Rousset, A. Termier

LIMSI (CNRS)
BP 133, Orsay Cedex, 91403, France
folch,habert,jardino@limsi.fr

LRI (CNRS-U. Paris XI), INRIA-Futurs (gemo team)
Université Paris XI, Orsay Cedex, 91405, France
pernelle,rousset,termier@lri.fr

Abstract

Extensible Markup Language (XML) is playing an increasingly important role in the exchange of a wide variety of data on the Web and elsewhere. It is a simple, very flexible text format, used to annotate data by means of markup. XML documents can be checked for syntactic well-formedness and semantic coherence through DTD and schema validation which makes their processing easier. In particular, data with nested structure can be easily represented with embedded tags. This structured representation should be used in information retrieval models which take structure into account. As such, it is meta-data and therefore a contribution to the Semantic Web. However, nowadays, there exists huge quantities of raw texts and the issue is how to find an easy way to provide these texts with sensible XML structure.
Here we present an automatic method to extract tree structure from raw texts. This work has been supported by the Paris XI University (BQR2002 project, Paris-XI University).

Introduction

The problem of structure extraction from a heterogeneous corpus of XML documents studied in (Termier et al, 2002) relies on discovering frequent tree patterns.
The problem that is addressed in this paper is quite different since its goal is to build tree structures from unstructured documents. The unsupervised clustering technique (based on Galois lattices) implemented in the Zoom system (Pernelle et al, 2002) requires to be applicable that the data can be described in function of a rather small subset of fixed features, which is not possible for textual data.
The database community has developed several powerful query languages for XML data, like Xquery which is the W3C query language or its navigational fragment Xpath (Clark and DeRose 1999). Those languages enable focusing search to subtrees characterized by tag paths or tree patterns, thus providing answers that are much more precise than full text search based on keywords. Making explicit tree structures in documents would make possible to benefit from the power of those query languages.

Unsupervised hierarchical clusterings seem the best way to bring to the fore a hierarchy in raw texts. These methods are commonly developed through commercial and free softwares (R project for Statistical Computing) and presented in several books (Duda and Hart 1973, Celeux et al, 1989, Kaufman and Rousseeuw 1990, Lebart et al, 1995). Actually, they have drawbacks which limit their use. The bottom-up method is well-adapted for small or medium-sized data, but it is a space and time consuming process. The advantage of the top-down method is speed but the arity of the nodes has to be specified in advance. In most of the cases, the arity is fixed to 2, and a binary tree is created. In both cases, each level in the resulting tree depends on the other ones. The issue is : does this tree reflect the true structure of the data?
Here, we propose to extensively apply an unsupervised partitioning algorithm to the data (texts) in order to highlight a data-based tree. However, it is worthwhile to notice that this method is generic and can be applied to any type of statistical data.

Structure extraction

We create a set of quasi-optimal and independent partitions of the texts with the clustering algorithm described below. Each partition corresponds to a specific number of clusters, ranging from 2 to K. We obtain K-1 partitions and we search among these ones the partition in which clusters remain stable. These stable clusters serve as anchors to build the hierarchy.

Clustering

A k-means like algorithm is used to automatically group texts in k clusters, details are given in (Jardino 2000).
In brief, texts are considered as bags of words and are represented by vectors whose components are the relative frequencies of the words in the texts. Texts are grouped in clusters represented by their centroids. The search of the optimal clustering is performed with an iterative process, according to an entropy-based criterion (Cover 1991).

Partitions

Firstly, a quasi-optimal partition of the text in two clusters is performed with the k-means like algorithm. At the beginning of this first clustering, the texts are grouped into one cluster. For the next processes, each clustering is initialized with the results of the preceding clustering : the clustering of the chunks in k clusters is initialized with the results of the clustering in k-1 clusters. Only a small bias is created by this procedure because the clustering process re-arrange the chunks randomly. We have chosen this protocole because it makes easier the visualisation of the paths between the stable clusters. The optimal partitions in 2, ..., k, ..., K clusters are respectively labelled P2, ..., Pk, ..., PK.
We draw (figure 1) an oriented graph whose nodes are the clusters and whose edges are the paths of the chunks in the successive partitions. There are K layers in the lattice, and a maximum of K! edges is possible between the clusters in the successive layers.

Tree search

We search the partition Pk in which clusters are stable. One cluster is said stable if its items (texts) are issued from only one cluster of the partition Pk-1. If this partition exists, it is possible to extract a tree from the graph by merging undifferentiated clusters in the partitions and clipping paths between identical clusters (see below).

Experiments

Corpus, segmentation

The corpus is a set of 1,100 touring information leaflets whose descriptive part has been selected. The 1,100 resulting texts correspond to about 333,000 words with a vocabulary size equal to 6,300 words. The average length of the texts is 295 words. These texts have a special syntax without any verb, they contain heterogeneous informations, separated by punctuation marks. We have used these marks to split the texts in chunks. Among the different marks we have chosen the point which insures the flatest distribution of the chunk lengths, we obtain 4,700 different chunks whose average length is 20 words.

Extraction of the hierarchy

Partitions of the chunks have been performed from P2 to P10. The associated graph is drawn on the figure 1.

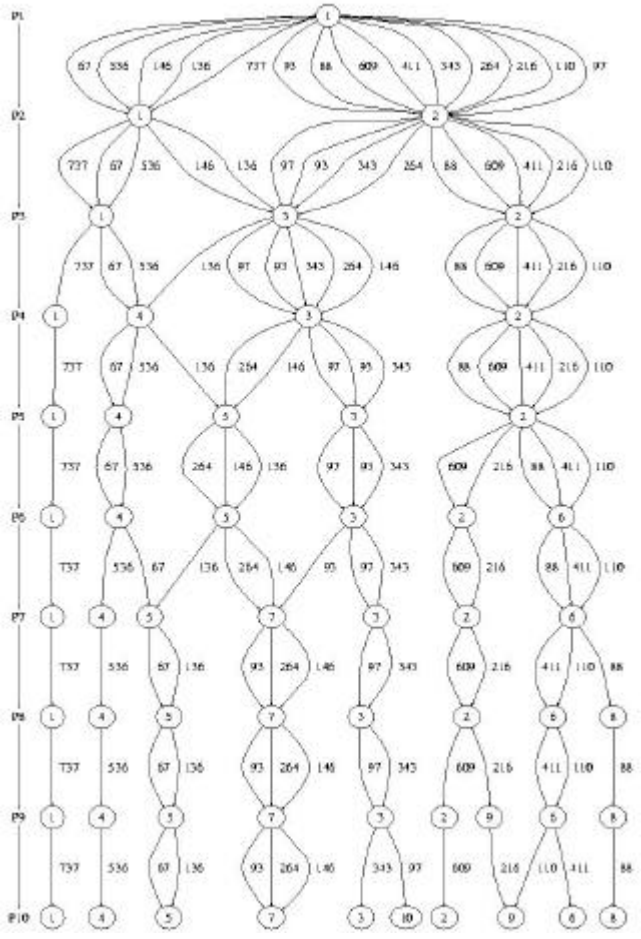

Figure 1: Oriented graph of the chunks in the partitions from 2 to 10 clusters. The values on the edges correspond to the number of chunks going down from P1 to P10.

It represents 75% of the data. We observe that the partition 8 is stable: each cluster is issued from only one cluster of the partition P7. So we cut the graph at this level and we consider the bottom-up paths in order to build a tree. We merge undifferenciated clusters. The clusters 3,4,5,7 of P7, 3,4,5 of P6 and P5, 3,4 of P4 correspond to a single node, we choose to label it with the labels of the clusters in the partition close to P1, that is to say "3,4 (P4)". The cluster 1 is stable from P4 to P8, we clip its branch under P4 and get the node "1 (P4)"and so on. We obtained an unbalanced tree (Jardino 2004), a partial view of which is given on the figure 2. The important point is that a part of the nodes of the tree are not the original clusters but the merging of several of them.

Hierarchy in a XML format

The projection of the tree structure on the chunks gives the XML embedding structure of the documents. Many projections are possible depending on the following choice : do we consider that two successive chunks that belongs to the same cluster have to appear under the same label or do we have to repeat the label and keep the informations separated. In fact, a very fine semantic analysis is often needed to decide. Assuming k levels in the tree and N items (chunks) in the tree, $(k+1)^{(N-1)}$ trees structures are possible in the worst case. So, we have chosen to encourage the recall rate and to aggregate chunks of the same label. The induced XML file is then a hierarchical database adapted to more fine-grained queries than key-words queries.

Labelling

The clustering process gives clusters which are labelled by numbers. In order to give sensible tags to the clusters we have extracted for each cluster a few texts which are near the virtual centroid of the cluster and we have reinforced this information with the most frequent words which appear mainly in the cluster (discriminant words). With this information, one of us has given sensible tags.

The cluster which contains informations about leisure and catering equipment is labelled *equipment*. The cluster which concerns options, promotions and extra charges is labelled *options.* The cluster about the global description of a residence is labelled *environment* and the cluster about board and lodging, basic rates and program is labelled *basic.* The more general levels of the hierarchy are labelled considering the label of their sons. The level 1 of the hierarchy fortunately aggregate *equipment* with *environment* and *options* with *basic*. The first one is called *place-desc* and the last one *rates&program.* The final hierarchy presented in figure 2 has been used to structure the corpus.

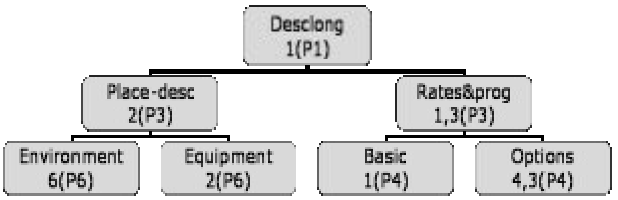

Figure 2: The labelled hierarchy with sensible tags and cluster affiliations

Evaluation

Two tests have been performed on this corpus: one of them was designed to evaluate the quality of the clustering, the other one to evaluate whether the structure improves the information retrieval.

Test 1 : a test corpus with 102 sentences has been formed in order to evaluate the hierarchy and its labelling. The sentences are chosen at random but the weight of each cluster is respected. Each sentence has been manually tagged using the list of the last level tags and their description. Then, we have compared the clusters created by the user and the unsupervised clusters. The recall and the precision rates for each cluster of the hierarchy are shown in table 1. Note that the recall and the precision of the clusters *place-desc* and *rates&program* have been calculated considering the kind of errors commited by the system at level 2.

Cluster	Precision	Recall
equipment	80,95%	85,00%
options	87,50%	68,29%
environment	48,40%	94,12%
basic	81,25%	54,16%
place-desc	66,67%	97,30%
rates&program	97,92%	72,31%

Table 1. Precision and recall

The clusters we found seem interesting. Besides, the clusters *place-desc* and *rates&program* are a good way to generalize the four clusters of level 2. Their recall rate is really higher than the recall of the two clusters they generalize. Actually, we have noted that an information which is wrongly aggregated in one cluster often concerns the cluster which have the same father in the hierarchy.

Test 2 : In order to test the retrieval performance of the structured documents, we have compared queries made on unstructured texts with queries made on the texts structured with the help of the hierarchy. In the first case, we search for travels such that the text contains a list of given keywords. In the case of more structured queries, we search for travels such that there exist a text of a given tag that contains a list of given keywords. The queries are formulated using Xpath, a language for addressing parts of an XML document. For each query, the hierarchical representation improves the precision of the returned document. For example, if the user wants to find the travels where a lagoon is situated in the environment of the hotel or the residence, our system finds 19 travels where the keyword lagoon appears in the *environment* tag whereas 62 travels are given by the non structured data. The excedentary leaflets correspond to the lagoon swimming pools, that is an information which concerns the *equipment* tag. Of course, not all the answers producted by the more structured query are relevant but the precision is up 23%. In the same way, if the user wants to find the travels where the destination is Paris, the precision rate is 33% using the tag environment whereas the precision rate is only 0.64 % if the unstructured data are used.
Actually, numerous trips are leaving from Paris and this information is described inside the tag *rates&program*.

Visualisation

We used a 2D visualization software 'Treemaps' (http://www.cs.umd.edu/hcil/treemap/), developed by Ben Shneiderman at the Human-Computer Interaction Laboratory (HCIL) of the University of Maryland [1] , as a visual aid to help us evaluate whether the induced XML structure improves information retrieval.
The Treemaps visualization method displays hierarchical data in a 2D space-constrained layout. In this layout, hierarchical information is represented as nested rectangles, in which nodes whose attributes have greater weight are allocated more space. The attributes that determine space allocation can be chosen by the user. In addition, users can also choose a second attribute that determines the colouring of the nodes in the display. Thus for instance we can define the size of nodes depending on the chunk entropy, and colour them on the basis of their cluster at a certain partitionning level.
We have modified the Treemaps software provided to us by the University of Maryland in order to adapt it specifically to the filtering of XML data by means of XPath expressions. The evaluation of an XPath expression selects a set of nodes from a XML document on the basis of constraints on the hierarchical structure of the XML document, the attribute values attached to XML elements as well as fulltext constraints. These nodes are highlighted in the Treemaps display.
Also, we made visible the text of the leaf nodes in the XML hierarchy, (i.e. the text content of each chunk) through a pop-up which is displayed when the mouse passes over a node in the 2D layout (Figure 3). Accessing the chunk's text content is crucial indeed to interpreting and labelling the clusters of chunks.
The Treemaps display enables rapid comparison of the number of nodes selected by different XPath queries and thus allows us to visually evaluate the precision and recall of XPath queries. We can thus compare the precision of queries that express constraints on the induced XML structure as opposed to those queries that only express fulltext constraints. Figure 3 shows the result of the XPath -unstructured query on the word "lagon" (nodes selected by the query are in white, the rest are greyed).

Conclusion

We have shown how to extract a latent struture in textual data: first, successive partitions of the texts are performed with an unsupervised clustering algorithm, then stable clusters are detected. These clusters serve as anchors to create a tree which is the image of the latent structure of the data. The method has been applied to a touristic information corpus and tests have been performed to evaluate both clustering quality and information retrieval performances. The results are encouraging. As the method is generic, it would permits to treate all kinds of corpus, in particular very large corpora.

[1] We are very grateful to Catherine Plaisant (HCIL-University of Maryland) for providing us with the source code of the Treemaps program.

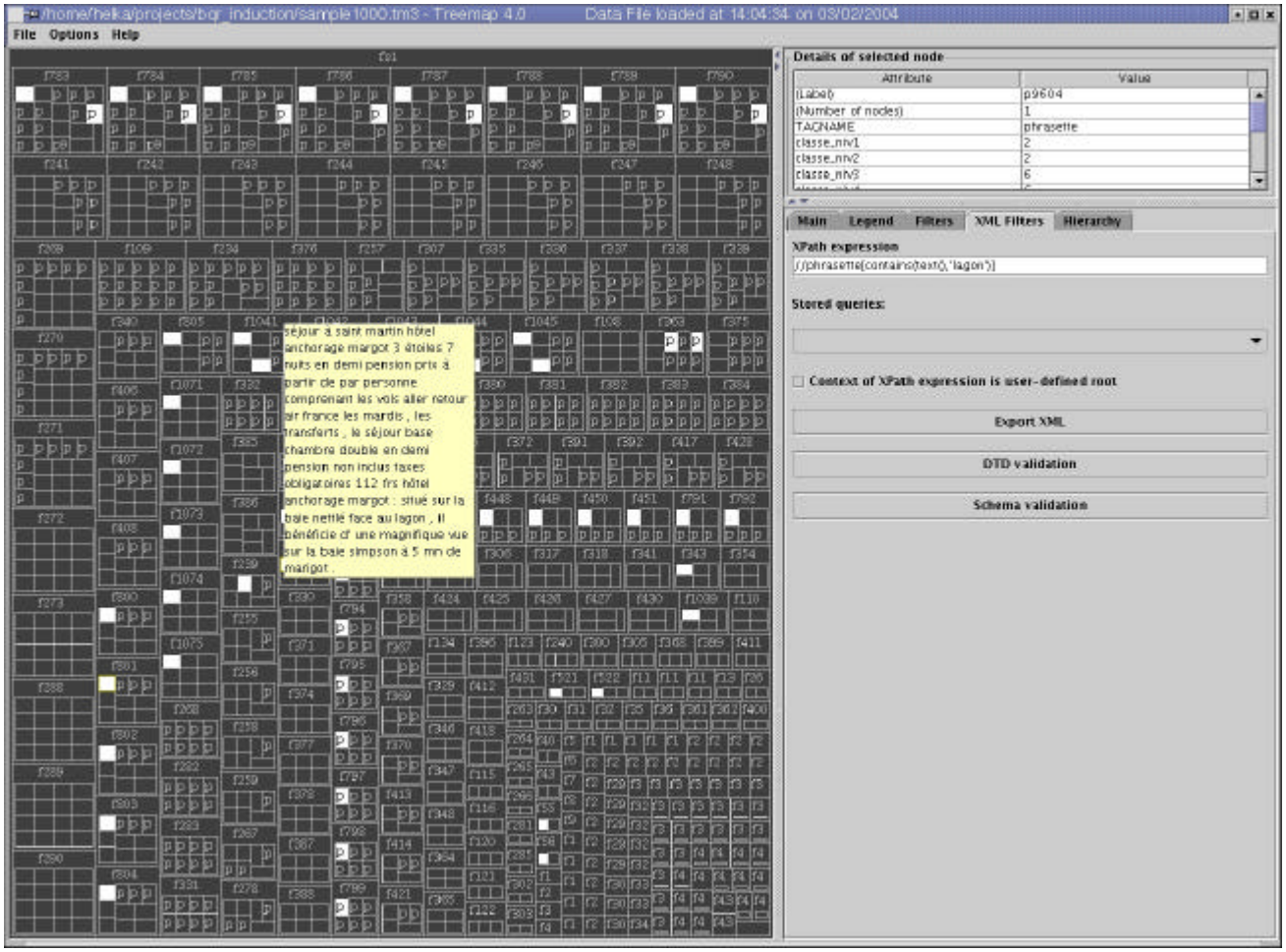

Figure 3: Treemaps display of the leaflets, result of the Xpath query on the word "lagoon"

References

Celeux G., Diday E., Govaert G., Lechevallier Y., Ralambondrainy H. (1989). *Classification automatique des données*. Dunod.

J. Clark and S. DeRose (eds) (1999). XML Path Language (XPath). 3C Recommandation, http://www.w3c.org/TR/xpath. Xquery 1.0: an XML Query Language, http://www.w3.org/TR/xquery/

Cover T. and Thomas J. (1991). *Elements of Information Theory*. Wiley & sons.

Duda R.O. and Hart P.E. (1973). *Pattern classification and Scene Analysis*. Wiley & sons.

Folch H., Habert B., Lahlou S. (2000). Navigable Topic Maps for overlaying multiple acquired classifications. MIT PRESS, Markup Langagues: Theory & Practice, 2 (3), pp. 269-280.

INEX, http://qmir.dcs.qmw.ac.uk/INEX

Jardino M. (2000). Unsupervised non-hierarchical entropy-based clustering. In Data Analysis, Classification and Related Methods. H.-H.Bock, W. Gaul, M. Shader Eds. Springer.

Jardino M. (2004). Recherche de structures latentes dans des partitions de «textes» de 2 à K classes. JADT2004.

Lebart L., Morineau A., Piron M. (1995). Statistique exploratoire multidimensionnelle. Dunod.

Kaufman L., Rousseeuw P.J. (1990). *Finding groups in data*. Wiley & sons.

Pernelle N., Rousset M.C., Soldano H. and Ventos V. (2002). Zoom: a nested Galois lattices-based system for conceptual clustering. Journal on Experimental and Theoretical Artificial Intelligence, vol.14, pp.157-187.

R, project for Statistical Computing, http://www.r-project.org/

Shneiderman, B. (1992) Tree visualization with treemaps: a 2-d space-filling approach. ACM Transactions on Graphics, vol. 11, 1, p.92-99.

Termier A., Rousset M.C. and Sebag M. (2002). TreeFinder: a First Step towards XML Data Mining. Proceedings of the International Conference on Data Mining ICDM 02.