Knowledge-free Table Summarization

Dino Ienco[1,2], Yoann Pitarch[3], Pascal Poncelet[1,2], and Maguelonne Teisseire[1,2]

[1] IRSTEA, UMR TETIS, Montpellier, France
{dino.ienco,maguelonne.teisseire}@irstea.fr
[2] LIRMM CNRS Montpellier, France
pascal.poncelet@lirmm.fr
[3] LIRIS CNRS Lyon , France
yoann.pitarch@liris.cnrs.fr

Abstract. Considering relational tables as the object of analysis, methods to summarize them can help the analyst to have a starting point to explore the data. Typically, table summarization aims at producing an informative data summary through the use of metadata supplied by attribute taxonomies. Nevertheless, such a hierarchical knowledge is not always available or may even be inadequate when existing. To overcome these limitations, we propose a new framework, named **cTabSum**, to automatically generate attribute value taxonomies and directly perform table summarization based on its own content. Our innovative approach considers a relational table as input and proceeds in a two-step way. First, a taxonomy for each attribute is extracted. Second, a new table summarization algorithm exploits the automatic generated taxonomies. An information theory measure is used to guide the summarization process. Associated with the new algorithm we also develop a prototype. Interestingly, our prototype incorporates some additional features to help the user familiarizing with the data: (i) the resulting summarized table produced by **cTabSum** can be used as recommended starting point to browse the data; (ii) some very easy-to-understand charts allow to visualize how taxonomies have been so built; (iii) finally, standard OLAP operators, i.e. drill-down and roll-up, have been implemented to easily navigate within the data set. In addition we also supply an objective evaluation of our table summarization strategy over real data.

1 Introduction

Nowadays, modern technologies allow to collect huge amount of raw but potentially very knowledgable data which are available at a very low level of granularity, e.g, sensor data, web logs. When facing this very detailed information the domain expert might experience big trouble determining from where he/she can start exploring these data. Actually, this data exploration is typically the first step in any analysis process. Nevertheless, when the available quantity of data is too abundant or the user is not so expert in the domain, this first step is often problematic and automatic approaches can help to obtain some insights about the way to effectively explore the data [**?**]. For instance, Explorative Data

Mining (EDM) tools help the user in preliminary analysis giving him/her the possibility to have a first meaningful view on the data. In this direction, it should be noted that two orthogonal research problems might coexist. Assuming a relational table setting, i.e., lines correspond to tuples whereas rows correspond to attributes/dimensions, the former problem is related to deal with the curse of dimensionality by the means of dimensionality reduction or aggregation [**?**], whereas the latter problem deals with reducing the number of tuples while providing the user with still a very representative view of his/her data. In this paper, we address the latter, i.e., the *tuple reduction* problem.

One way to manually deal with this *tuple reduction* problem is using data warehouses coupled with OLAP engines [**?**]. These interactive tools allow the analyst to aggregate and navigate through data by the mean of the drill-down and roll-up operators that exploit, when available, the attribute hierarchies/taxonomies[4]. OLAP engines demand human supervision while an automatic way for addressing this problem is supplied by the table summarization process (TSP) [**?**].

The table summarization process (TSP) provides an aggregated representation of a relation table and thus helps the user familiarizing with the data. Typically, TSP is particularly well adapted to the rapid development of mobile devices, e.g., smart phones or tablets, since it enables visualizing data respecting the screen size constraints imposed by the new devices generation [**?**]. To cope with this issue, TSP can be exploited from smartphone devices to supply synopsis (or summary) of relational data. In this work we focus our research on the second issue and thus propose a new innovative table summarization algorithm.

Our work is essentially motivated by the following observation: data often lacks of associated metadata information, e.g., attribute taxonomies. This lack drastically limits the panel of possible approaches (OLAP, TSP) to use to reduce the number of tuples. Indeed, both OLAP and existing TSP techniques assume attribute hierarchies to be available. This assumption is clearly a strong constraint because most of the times such a hierarchical knowledge is not available or may even be inadequate when existing. On the other hand, using inappropriate hierarchy can provide useless results since they can be too general and bias the OLAP and TSP results. Thus, proposing techniques that are able to meaningfully summarize huge quantities of data without leveraging any other knowledge than the one provided by the data might be very useful for analysis purpose.

Contrary to the previous techniques, our framework, called **cTabSum**, relaxes this constraint producing table summaries without needing any additional information. The main advantage of **cTabSum** lies in its ability to automatically generate contextual Attribute Values Taxonomies (cAVTs) and to use them to automatically summarize the data. This original feature also allows our approach to be used over relation tables where metadata information are too general or do not fit the domain of investigation. An example of application can be represented by the visualization, over mobile device, of query results. **cTabSum**

[4] In this paper we will use these two terms indistinguishly.

can automatically aggregate query results exploiting data dependency. After the performed aggregation, the summarized table can be easily visualized on a smartphone screen. The final result of **cTabSum** is a summarized table that can be considered as a recommended aggregation of the data. It can also be considered as an entry point from which the user can start browsing the data using drill-down and roll-up operators. The main contributions of our work are the following:

- A general table summarization approach that can be applied over data table where no metadata are available or metadata are too general for the considered domain;
- The final result can be exploited as a starting point to browse and navigate through the data;
- A prototype that integrates the **cTabSum** algorithm in an interactive and easy-to-use environment. Drill-Down and Roll-up operators are implemented to navigate data starting from the recommended summarization.

The rest of the paper is organized as follows. Section 2 presents the related work and positions our contributions w.r.t. the state of the art. In Section 3 we describe our proposal to produce TSP without any kind of metadata information. An experimental evaluation over real data is carried out in Section 4. In Section 5 we present the prototype built over the **cTabSum** algorithm and how the user can interact with the resulting summarized table. We also supply an illustrative example with the purpose to realize a qualitative evaluation of our approach. Section 6 concludes and describe some possible future works.

2 Related Work

A first approach proposed in the context of table summarization was presented in [**?**]. This approach creates and maintains table summaries through row and column reductions. To reduce the number of rows, the algorithm first partitions the original table into groups based on one or more attribute values of the table, and then collapses each group of rows into a single row relying on the available metadata, such as the concept hierarchy. In [**?**] the SaintEtiQ is proposed. This system computes and incrementally maintains a hierarchically arranged set of summaries of the input table. SaintEtiQ uses background knowledge (i.e., metadata) to support these summaries. [**?**] proposed to work at the metadata level to improve the final summarization and speed-up the whole process. The strategy is based on a pre-processing of the original concept taxonomy in order to obtain a more compact hierarchy. Once the reduced metadata is available it is employed to perform the table summarization task. As we can observe, all the previous algorithms exploit some metadata knowledge that compulsoliry need to be supplied by the user. On the contrary, our approach is able to perform and complete the process of table summarization without asking any additional information to the expert. This characteristic allows **cTabSum** being much more flexible and usable when no background knowledge is available.

3 cTabSum

In this section we first introduce preliminary notations and then describe in details the **cTabSum** framework. We define a table T as a set of tuples, i.e., $T = \{t_1, t_2, ..., t_n\}$. Each t_i is described over a set of attributes $A_1, \ldots, A_m$, i.e., $t_i \in (dom(A_1) \times dom(A_2) \times ... \times dom(A_m))$. Given $1 \leq i \leq n$ and $1 \leq j \leq m$, T_{ij} denotes the value of the attribute A_j in the i^{th} tuple of T. Given an attribute A_j, V is a partition over the values of that attribute such that $\forall_{v \in V}\ v \subset A_j$ and $\forall_{v,w \in V}\ v \cap w = \emptyset$. The partition V induces a generalization of the attribute A_j. More precisely, given a value $a_{jk} \in dom(A_j)$ (a_{jk} is the k-th value of the attribute A_j), it exists a value v s.t. $a_{jk} \in v$. For instance, given an attribute taxonomy over A_j, the ancestor of an attribute values can be seen as a generalization of it. The objective of the TSP is to find a summarized table $s(T) = \{g(t_1), g(t_2), ..., g(t_l)\}$ s.t. each $g(t_b)$ (generalized tuple) cover at least k tuples of the original table T and there are no overlap among the tuple set covered by generalized tuples.

3.1 Proposed Approach

Essentially the proposed strategy proceeds in two main steps. The first step consists in building the cAVTs (one for each attribute A_j over which T is defined). Leveraging data mining techniques, we can extract this meta-information starting from the data [**?**]. The second step performs the table summarization strategy. The algorithm uses the previously calculated cAVTs to generalize the data. To do so, a new approach that exploits a criterion coming from information theory to evaluate the quality of the possible solutions in the search space is proposed. Similarly to [**?**], **cTabSum** also requires a parameter k but it avoids any metadata information (taxonomies) as input.

The TSP extracts a summarized table with the constraint that each generalized tuple must cover at least k tuples of the original table, where k is a user-defined parameter. The table summarization task can be related to the k-anonymity problem[**?**]. While the algorithmic approaches could be similar, the final results are different in spirit. TSP supplies an informative compression of data to allow exploratory analysis in contrast to the k-anonymity purpose which aims at hiding some sensitive information. Another difference is that table summarization is performed for all the attribute of the data table while k-anonymization involves only the set of attribute composed by quasi-identifiers (sensible attributes).

3.2 Table-Summarization Quality Criterion

The TSP task can be seen as a search problem in which we want to find, if possible, the best compression of the original table taking into account the constraint k. This best compression also implies to have the minimum information loss w.r.t the original data. Thus, when facing with a set of possible intermediate solutions, it is needed to understand which one should be chosen. In the field of data compression, information theory measures are normally used to evaluate,

given a model, the gain in compression (and information loss) w.r.t. the uncompressed data [**?**]. As our final goal is to obtain a summarization of the original table, these kind of measures can be very suitable. Particularly, to evaluate how much information is lost during the summarization process, we employ the non-uniform entropy [**?**]. The non-entropy measure, that we call $InfoLoss(T, s(T))$, is defined as follows:

$$InfoLoss(T, s(T)) = \sum_{i=1}^{n} \sum_{j=1}^{m} -\log Pr(T_{ij} = a_{jk} | s(T)_{ij} = v) \quad (1)$$

where

$$Pr(T_{ij} = a_{jk} | s(T)_{ij} = v) = \frac{\#\{1 \leq i \leq n : T_{ij} = a_{jk}\}}{\#\{1 \leq i \leq n : T_{ij} \in v\}}$$

The conditional probability $Pr(T_{ij} = a_{jk} | s(T)_{ij} = v)$ indicates how the subset of original values of A_j are distributed w.r.t. their generalization induced by partition v. The non uniform entropy measure (w.r.t. the classical entropy measure) helps to better considering the distribution of the values belonging to the same generalization.

3.3 Building CAVTs

An important step of our strategy is the automatic construction of the cAVTs. To structure attribute values in taxonomies we exploit the approach proposed in [**?**]. This technique allows to extract distances between each pair of values of the same categorical attribute. Most in detail, given a categorical attribute A_j, over which the database D is defined, it selects a set of other attribute strictly related to it using correlation measure. This set of attribute is called context. After that, it uses the distribution of the values of attribute A_j w.r.t. the attributes in the context to infer a distance matrix that represents the distances between each pair of values $v_{jk}, v_{jl} \in A_j$, where v_{jk} is the k-th value of attribute A_j. Once the point-wise distance matrix for A_j is available, it is used as input for standard agglomerative hierarchical clustering[**?**] that produces the final cAVTs[**?**]. This approach is very useful when the original hierarchies are not available or to compare the induced taxonomies with the original ones understanding which one fit the actual analysis. As the TSP process requires hierarchies to summarize the original data, the cAVTs always supply this metadata knowledge. In particular, given an attribute, any cuts of the corresponding cAVT results in a partition of the domain of that attribute. Each of these partitions represents a possible generalization that can be used in the summarization process.

3.4 Constructing the Summary

At this point we are able to extract attribute taxonomies directly from data (cAVTs) and evaluate how far is a summary w.r.t. the original data (InfoLoss). Now we need to design a strategy to navigate the search space defined over all

the possible solutions. In this direction we design an algorithm for TSP following the general idea presented in [**?**]. There are three big differences between the proposed method and the one described in [**?**]:

1 **cTabSum** supplies a summary of the original data for exploratory purpose while in [**?**] the final goal is the anonymization of the data following the k-anonymity philosophy [**?**].
2 In [**?**] the goal is to preserve classification accuracy and for this reason the class variable is used to guide the privacy preservation process. **cTabSum** is designed for a totally unsupervised scenario where no class information is available and the final goal is to compress the original data. To implement this important difference, instead of the Information Gain used in the privacy preserving strategy we minimize the Information Loss criteria based on Non Uniform Entropy.
3 **cTabSum** does not require any metadata because it directly mines this knowledge from the data relaxing the strong assumption on the availability of such kind of information.

The general process is presented in Algorithm 1. It takes as input a value k and returns a set of generalized tuples that describes the original table. The parameter k indicates the minimum number of original tuples that need to be covered by each generalized tuple in the final result. The search strategy is performed by a top-down navigation over the space of possible solutions. The top-down search starts from the top of all the attribute value taxonomies (the coarsest granularity levels) and, at each step, tries to specialize and expand only one attribute taxonomy with the function $bestExpansion(T, H, x)$. This function chooses to expand the cAVT that produces the minimum loss of information evaluated by the Formula 1. Most in detail, for each attribute A_j the following steps are performed: (i) generate all the possible expansions starting from the actual frontier of the cAVT, (ii) choose the best expansion that minimize the measure in Formula 1.

As the process is performed over all the attributes simultaneosuly, $bestExpansion(T, H, x)$ can expand any of the taxonomies at any node. This means that in the final summarized table, for the same attribute, we can have a generalization (partition) at different levels of granularity. Once that the actual expansion y is chosen we evaluate both: the Information Loss w.r.t. the actual best solution and the constraint supplied by k. If both conditions are satisfied, y is marked as new best solution. At the end the algorithm returns the solution that minimize the Information Loss and, at the same time, the constraint k.

cTabSum is implemented in a tool that has the same name and it is public available at the url address: `http://www.lirmm.fr/~ienco/Dino_Ienco_Home_Page/cTabSum.html`.

4 Experiments

In this section we draw some experiments to analyze the performance of our approach using real world data table. The approach has been implemented in Java

Algorithm 1: $cTabSum(T, k)$

```
/* H is the set of cAVTs induced from T                                    */
H = extractCAVTs(T);
actualFrontiers = ∅;
Initialize x to the Summarization using the top value of each taxonomies in H;
actualFrontiers = actualFrontiers ∪ x;
InfLoss = Score(x);
result = x;
forall the x ∈ actualCuts do
    y = bestExpansion(T, H, x);
    actualFrontiers = actualFrontiers \ x;
    actualInfLoss = Score(y);
    if (actualInfLoss ≤ InfLoss AND y is valid w.r.t. k) then
        InfLoss = actualInfLoss;
        actualFrontiers = actualFrontiers ∪ y;
        result = y;
    end
end
return result;
```

and we carried out experiments on Mac OS X 10.6.8 with 4Gb of RAM and an i7 2.2GHz processor. Actually this task shares similarities to the k-anonymization problem, in which each tuple of the anonymized table needs to represent at least k tuples of the original table. For this reason we used the algorithm MinGen presented in [**?**] as competitor to evaluate the performance of **cTabSum**. This algorithm was originally developed for the k-anonymity problem. We couple MinGen with the original available taxonomies. The algorithm needs two parameters. The first parameter k is the minimum number of tuples indistinguishable over the set of sensitive attributes (quasi-identifiers) [**?**]. The quasi-identifiers are the attributes for which the generalization process is performed (and the attribute taxonomies is demanded). The second parameter is the suppression coefficient *sc* standing for the maximum number of tuples (in percentage w.r.t. the size of the table) that the algorithm suppresses to obtain a k-anonymized table. This parameter is not relevant for the table summarization task and it has thus been set to 0. To evaluate the performance we use the *Adult* dataset[5]. It is based on census data and has been widely used to evaluate classification and k-anonymization algorithms. Here, the same settings as in [**?**] are used. We obtain a dataset of 8 attributes. The attribute *age* is discretized in 8 equal-size bins. Starting from this dataset we sample 10% of the original dataset obtaining 4 522 tuples. As original hierarchies we use the taxonomies supplied in [**?**]. In our experiment we range the k parameter from 40 to 400 at step of 40. To evaluate the quality of the compressed table we adopt the non-uniform entropy measure (Formula 1). The idea is that a good k-anonymization (for this reason also a good summary of an original table) is the one that minimizes the Information Loss respecting the given constraints. As **cTabSum** generates attribute taxonomies directly from the data, a satisfactory result could be to obtain comparable performances compared to the ones involving the original hierarchies.

[5] http://archive.ics.uci.edu/ml/

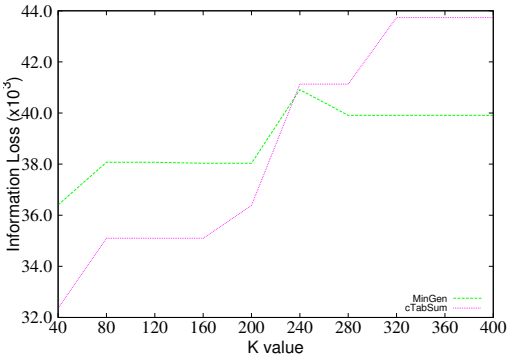


Fig. 1. Information Loss of **cTabSum** and *MinGen* over the *Adult* dataset

In Figure 1 we observe the behaviors of the different approaches. Generally, we can state that the two approaches have comparable performances. This is a good point for **cTabSum** that automatically generates attribute taxonomies from data. In particular for values of k lesser than 240 **cTabSum** obtains better results in terms of Information Loss. This value of 240 influences seriously the final summarization because for values above this threshold both algorithms return summaries of only 4 generalized tuples to represent the original data table. From an user point of view, this result can be useless due to the fact that it compresses to much the original information. With values lesser than 240, both approaches return more reasonable summarization composed at least of 8 tuples. These results can represent more reasonable starting points for an initial explorative analysis of the data.

4.1 Assessing Summarization via Swap Randomization

In this subsection we evaluate how far the results of **cTabSum** are from randomness. A table summarization could be seen as a partition of the original tuples in groups that share common characteristic w.r.t. the attribute taxonomies. In this way TSP produces a clustering of the tuples constrained by metadata. Unfortunately we do not have any background knowledge to evaluate the quality of this partition. For this reason following the idea proposed in [**?**], we compare our results w.r.t randomized partition. Practically, given a table summarization result, each generalized tuple represents a cluster of the original ones. Starting from this clustering result, we swap at random, tuples between clusters obtaining random partition. In the random partitions the marginal distribution given by the original clustering are maintained. To evaluate the clustering solution we employ the ZIP function always available over any OS. In particular we zipped each cluster separately and we sum the size of all of them for a clustering solution in order to obtain a value for each value of k. The value is expressed in Kb. Low values indicate high compression rate that means a kind of structure inside the

partitions. We employ this kind of measure because because it naturally captures how good is the compression of the original table. The ZIP procedure also constitutes an easy way to evaluate this quantity. Figure 2 shows the results. For each value of k we generate 100 random clustering following the distribution induced by the result of **cTabSum** and we report the average value. We avoid to visualize the standard deviation because it is always smaller than 1kb. We can observe that the results produced by **cTabSum** always give a lower size of the final ZIP files, this underline how the clustering induced by **cTabSum** contains a structure that can be easily recognize by a compression algorithm as the one implemented by the ZIP function.

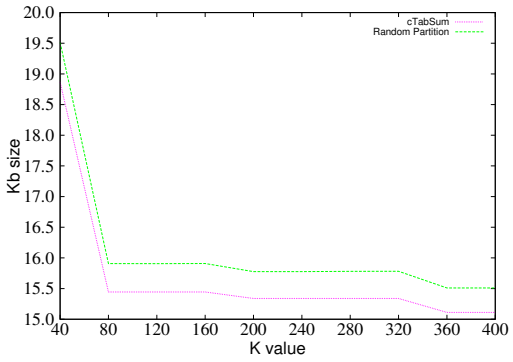
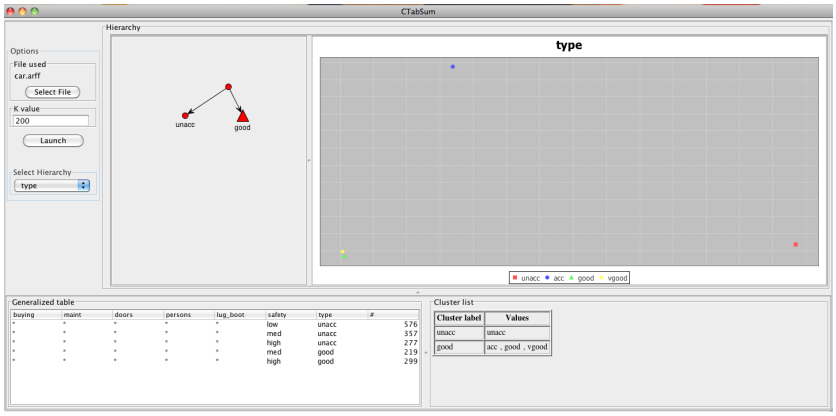


Fig. 2. Size in Kb of zip files comparing the group obtained by **cTabSum** and the randomization result

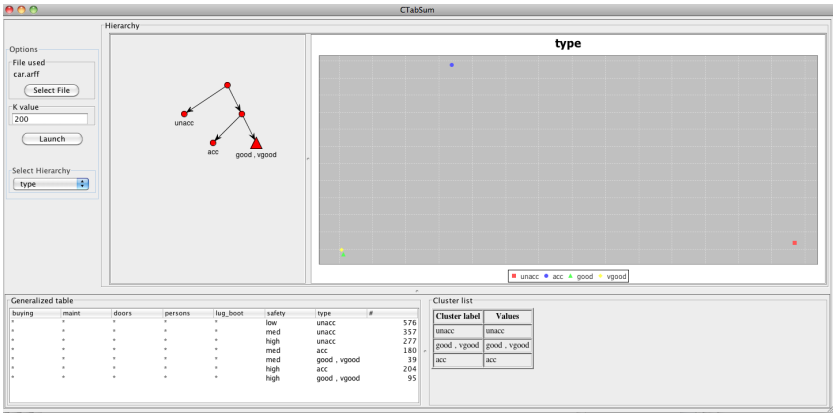
5 The cTabSum System

As we have obtained interesting results from an experimental point of view, our algorithm has been integrated in a prototype named **cTabSum**. **cTabSum** is a standalone software written in Java. It is based on several open source projects. First of all, it uses the *Weka Data Mining Library* to manage the data[1]. More precisely, our system requires a file in ARFF format (Attribute Relational File Format). In the GUI we integrate the *JFreeChart* Library[2] to produce and visualize 2D chart and the *JUNG* (Java Universal Network/Graph) library[3] to visualize attribute taxonomies. Figure 3(a) and 3(b) show a screenshot of **cTabSum**. It illustrates an example of result obtained by a TSP. It allows to choose a source file (in ARFF format) and to set the value of the parameter k. In this example the *Car* dataset was used with a parameter value, k, equals to 200. In this dataset each instance

[1] http://www.cs.waikato.ac.nz/ml/weka/
[2] http://www.jfree.org/jfreechart/
[3] http://jung.sourceforge.net/



(a)



(b)

Fig. 3. The **cTabSum** system.

is described over a set of discrete features (e.g., *buying*, *safety*, *type*, etc..). Once the TSP is launched, the final table summarization result is shown in the bottom part of the GUI. The result of the summarization process is displayed in table that contains all the original attributes plus an extra attribute, *Count*, to count how many original tuples are covered by each generalized tuple. The $*$ symbol indicates that the root level of the corresponding attribute taxonomy is used, i.e., all the values are grouped together. In the proposed example, the algorithm has found a solution that involves the specialization of only two attributes: *safety* and *type*. Starting from this result the user can interact with the data. The combo box in the top left part of the window allows to choose and investigate a particular attribute of the dataset. In the middle of the window, the cAVT can be visualized. In the tree visualization panel, two different types

of nodes coexist: triangle nodes represent group of two or more original values while circle nodes represent group of only one attribute value.

In this example, we observe that the taxonomy associated to the attribute *Type* is cut in a way to produce two groups of attributes: the left node with only one value, *unacc* (unacceptable), and the right node that contains a group of three values, *acc*, *good* and *vgood*. The actual taxonomy cut can be also visualized in the right bottom part of the interface in which each cluster is described with its representative plus the enumeration of the attribute values that it contains (clustering list). The user starts from this taxonomy cut to browse the hierarchy using drill-own and roll-up operators. These operations modify the displayed attribute taxonomy. The changes in the attribute taxonomy are propagated to all the other components in the GUI, i.e., *Generalized table* and *Cluster list*. In Figure3(b) the application displays the results after an user action over the attribute taxonomy. We can observe that the user has expanded the taxonomy *Type* thanks to a drill-down operation. It should be noted that this expansion impacts on both the panels *Generalized table* and *Cluster list* that now displays the current view of exploration. This means that the *Generalized table* panel visualizes the generalized tuples correspondingly to the actual cut of the different attribute taxonomies while the *Cluster list* shows the partition of leaves of the selected attribute with respect to the considered level of granularity.

Another feature implemented in **cTabSum** is shown in the right upper side. For each attribute, a 2D chart is plotted where the attribute values are positioned in a way that preserves the relationships existing in the original pointwise distance matrix. When a user selects a new attribute from the combo box, both the chart and the taxonomy are automatically updated to display information related to the newly selected attribute. As an additional feature supplied by our prototype, when there some attribute in the dataset are numerical, **cTabSum** exploits the pre-processing facility supplied by Weka and performs a discretization step deriving 10 equal-width bins for each numerical attribute. In this way our system is able to manage tables containing mixed attributes types. The **cTabSum** implementation is available at the URL: `http://www.lirmm.fr/~ienco/Dino_Ienco_Home_Page/cTabSum.html`.

6 Conclusion

The process of Table Summarization helps the analyst to have a first picture of the data. In our work a new algorithm to perform table summarization **cTabSum** is proposed. One of the main features of **cTabSum** consists in relaxing the strong assumption on the availability of attribute taxonomies. This is realized mining directly the attribute taxonomies from the data. Associated with the new algorithm we also developed a java prototype designed to fully exploit the knowledge available in the data. It uses **cTabSum** to aggregate the original data to supply (i) a more compact representation of them (ii) a starting point from which the user can navigate the table. More precisely, starting from the result of our system, the user can browse, interact and modify the data summary

navigating through the attribute taxonomies performing both drill-down and roll-up operations. In the future we plan to integrate **cTabSum** in open source OLAP project (like Mondrian) in order to supply a sort of recommendation to the final user to start his/her analysis. From an algorithmic point of view we want eliminate the use of parameter k replacing it with a lower bound concerning the minimum number of generalized tuples the final result may contain. Finally, the goal of our study is to summarize simple relational table, in which static information appears. In a next step, summarizing more complex data, e.g., data with temporal dimension, could be considered. In such a case, the generation of cAVTs will be different and much complicated and require further investigations.