A new type of metadata for querying data integration systems

Sonia Bergamaschi[1], Francesco Guerra[2], Mirko Orsini[1], and Claudio Sartori[3]
bergamaschi.sonia,guerra.francesco,orsini.mirko@unimore.it
claudio.sartori@unibo.it

[1] DII-Università di Modena e Reggio Emilia
via Vignolese 905, 41100 Modena
[2] DEA-Università di Modena e Reggio Emilia
v.le Berengario 51, 41100 Modena, Italy
[3] DEIS-Università di Bologna
Viale Risorgimento 2, 40136 Bologna, Italy

Abstract. Research on data integration has provided languages and systems able to guarantee an integrated intensional representation of a given set of data sources. A significant limitation common to most proposals is that only intensional knowledge is considered, with little or no consideration for extensional knowledge.
In this paper we propose a technique to enrich the intension of an attribute with a new sort of metadata: the "relevant values", extracted from the attribute values. Relevant values enrich schemata with domain knowledge; moreover they can be exploited by a user in the interactive process of creating/refining a query. The technique, fully implemented in a prototype, is automatic, independent of the attribute domain and it is based on data mining clustering techniques and emerging semantics from data values. It is parametrized with various metrics for similarity measures and is a viable tool for dealing with frequently changing sources.

1 Introduction

Integration of data from multiple sources is one of the main issues facing the database and artificial intelligence research communities. A common approach for integrating information sources is to build a mediated schema as a synthesis of them. By managing all the collected data in a common way, a mediated schema allows the user to pose a query according to a global perception of the handled information. A query over the mediated schema is translated into a set of sub-queries for the involved sources by means of automatic unfolding-rewriting operations taking into account the mediated and the sources schemata. Results from sub-queries are finally unified by data reconciliation techniques (see [9, 1] for an overview).

Research on data integration has provided languages and systems able to guarantee an integrated representation of a given set of data sources. A significant limitation common to most proposals is that only intensional knowledge is considered, with little or no consideration for extensional knowledge.

In this paper, we describe a technique for providing metadata related to attribute values. Such metadata represent a synthesized and meaningful information emerging

from the data. We call these metadata "relevant values" as they provide the users with a synthetic description of the values of the attribute which refer to by representing with a reduced number of values its domain. We claim that such metadata are useful for querying an integrated database, since integration puts together in the same global class a number of local *semantically similar* classes coming from different sources and a set of global attributes which generalize the local classes. Consequently, the name/description of a global class/global attribute is often generic and this fact could significantly limit the effectiveness of querying. Let us suppose, for instance, that the user has a good knowledge of a single source, say "S", and that she/he is interested in items whose global attribute "A" contains the word "x", as of terminology of source "S". The user could completely miss the fact that in source "T" the word "y" refers to a very similar concept, and therefore a query with target "x" would return only a partial result, w.r.t. the contents of the global class. Moreover, ignoring the values assumed by a global attribute may generate meaningless, too selective or empty queries. On the other hand, knowing all the data collected from a global class is infeasible for a user: databases contain large amount of data which a user cannot deal with. A metadata structure derived from an analysis of the attribute extension could be of great help in overcoming such limitation.

This work is done in the context of the MOMIS (Mediator envirOnment for Multiple Information Sources) project[4] [4], a framework to perform information extraction and integration from both structured and semi-structured data sources, plus a query management environment able to process incoming queries through the navigation of the mediated schema. The MOMIS integration process gives rise to a Global Virtual View (GVV) in the form of Global Classes and global attributes of the a set of data sources. In [3], we proposed a partial solution to the semantic enrichment of a GVV by providing a semantic annotation of all the Global Classes of the GVV with respect to the WordNet lexical database[5], and thus providing each term with a well-understood meaning. Relevant Values will semantically enrich a GVV, since they provide semantic information about the data sources the GVV refers to. Moreover, in [2] a first heuristic for calculating relevant values was described.

In this paper we improve the approach proposed in [2], by providing a flexible parametric technique to deal with string attributes. It is not a severe limitation, as: (1) data coming from web-site wrappers are generally represented as strings; (2) several techniques have been developed in literature for clustering numeric values where it is easy to define element orderings (see [8] for a survey). The method was implemented in a prototype called *RELEVANT* (RELEvant VAlue geNeraTor) we describe in section 3.

The outline of the paper is the following: next section defines the technique to elicit relevant values for a selected attribute, section 3 describe the implemented prototype and section 4 describes how relevant values may be exploited for querying data sources. Finally section 5 sketches out some conclusions and future works.

---

[4] See http://www.dbgroup.unimo.it for more publications about the project.

[5] http://wordnet.princeton.edu/

2 Eliciting Relevant Values from Data

There are several models for representing knowledge bases. Without loss of generality, let us refer to the concepts of MOMIS. The Global Virtual View built with MOMIS is composed of Global Classes, with Global Attributes (GA). Our goal is to extract the relevant values of a GA. Each relevant value is described by a relevant value name and a set of values of the attribute domain.

The idea is that analyzing an attribute domain, we may find values which may be clustered because *strongly related*. Providing a name to these clusters, we may refer to a relevant value name which encompasses a set of values. More formally, given a class C and one of its attributes At, a **relevant value** for it, rv^{At} is a pair $rv^{At} = \langle rvn^{At}, values^{At} \rangle$. rvn^{At} is the name of the relevant value set, while $values^{At}$ is the set of values referring to it.

Now we should answer two questions: how can we cluster the values of the domain in order to put together in a relevant value a set of values which are strongly related? How can we choose the relevant value names? The first question will be answered by means of clustering techniques, adapted to the problem on hand; the second will require the intervention of the designer, but we will provide, in section 3.4, an effective *assistant*.

Like most cluster tasks with non-numeric attributes, the problems are related to find an effective representation of the points (i.e. the attribute values) in a space, and to devise a suitable similarity function to be exploited by the clustering algorithm. The technique we propose builds a binary representation of the attribute values, and exploits two different kinds of measure to build some structure upon the flat set binary representation: 1) the *syntactic* similarity, mapping all the words of the attribute values in an abstract space, and defining a syntactic similarity function in such space; and 2) the *domination* measure, expressed by the root elements described later on. Such measures are automatically extracted: the manual annotation of each attribute value (e.g. with reference to a given ontology) would be a time-consuming and error-prone operation also discouraged by the high number of values and the update frequency.

The similarity measures we propose are then used by a clustering algorithm (in *RELEVANT* the user may generate both partitions and overlapped clusters). The clusters of values produced are so far called *sets of relevant values*. The user may balance the weight of the two different similarity measures.

2.1 The syntactic similarity

Terms related to the same object may have the same etymology and then share a common root: several similarity measures are based on this idea (e.g. the Levenshtein distance and the other metrics derived from it). In the same way, we may assume that related attribute values share terms. By means of this measure, we group different attribute values sharing common terms.

It is trivial to show that a term may be polysemous, i.e. it may be used in different attribute values with different meanings, especially in multi-word values. In our experience, the syntactic similarity alone could not be sufficient, but in conjunction with the similarities described below, it provides satisfactory results.

2.2 Domination: the root elements

A similarity measure may be extracted from the *Domination* relationships between the attribute values. Considering two attribute values a_1 and a_2, we say that a_1 dominates a_2 if a_1 is more "general" than a_2. Any partial order on attribute values could be used to define domination. On the basis of an analysis of several databases, we observed that it is frequent to have string domains with values composed of many words, also with abbreviations. We observed also that the same word, or group of words, may be further qualified (i.e. specialized) with multiple words in many ways. For example, the attribute describing a kind of production for a mechanical enterprise may contain the value "Mould" and the values "Mould ejectors, Mould engineering, ...". Thus, we approximate the domination between attribute values, a semantic property, with the *Contains* function, a syntactic property. *Contains* is a function based on string containment: $Contains(X,Y) = true$ iff $stem(X) \supseteq stem(Y)$, where X and Y are sets of words and *stem* is a *stemming operator* for words[6]. Then we say that Y dominates X if it is contained in X. The domination is a partial order and can be represented by an oriented graph. Let us say, for instance, that an edge goes from the dominating value (more general) to the dominated one (more specific). The integration designer should verify how much the graph represents the general notion of a value being "more general" than another, but in our experience the graph is usually sound.

Our idea is to exploit the domination for building clusters of values around *root elements*. A root element is an attribute value with only outgoing edges in the domination graph, and can be taken as a representative of the cluster composed by the nodes recursively touched by its outgoing edges.

3 The *RELEVANT* prototype

RELEVANT is a software tool for calculating relevant values. Giving as input a list of attribute values, *RELEVANT* generates a set of relevant values according to the designer selections. Figure 1 shows the *RELEVANT* functional architecture, which is organized into five blocks:

1. **Data pre-processing**: two binary representations of the values of an attribute are obtained with two matrices representing the different kinds of similarity measure.
2. **Similarity Computation**: the designer selects how to measure the similarity (metrics selection) and which kinds of similarity are used (the syntactic similarity, the domination measure or a combination of the two).
3. **Clustering technique selection**: we implement some clustering algorithms to compute the set of relevant values on the basis of the choices made at step 2.
4. **Name selection**: for each group of values defined in step 3, a name representative of all the values has to be identified.
5. **Validation**: we implemented some standard techniques to evaluate cluster quality. Additional work is necessary to go beyond the simple evaluation, so as to provide effective assistance to the designer in the critical task of parameter configuration.

[6] a standard operator in natural language processing.

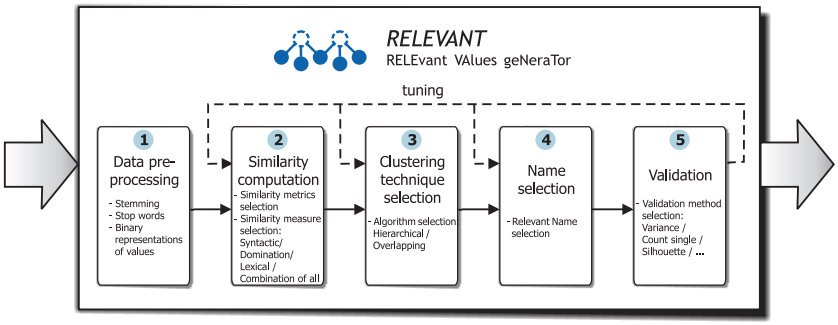

Fig. 1: The *RELEVANT* functional architecture

3.1 Step 1: Binary Representation of attribute values

The *RELEVANT* starting point is the creation of two binary matrices, according to the different measures introduced in section 2: MT and MTR.

The *Syntactic Matching Table* (MT) is a binary representation of all the values of an attribute At w.r.t. the universe of words considered (i.e. is the union of the words included in the extension of At). Notice that multi-word attributes contribute to the universe of words with multiple words. MT is typically sparse: for each row there is a number of elements different from zero that is equal to the number of words contained in the associated attribute, except for the stop–words.

The *Root Elements Matching Table* (MTR) shows the root elements associated to the attribute values: each column of the matrix is a root element and the rows are the attribute values.

3.2 Step 2: Similarity computation

Two tasks are executed in this step: the selection of the metrics for computing the similarity on the matrices created in the previous step and the computation of the affinity matrices AM and AMR derived from the matching tables MT, MTR respectively.

Concerning the first task, the tool implements some of the metrics commonly adopted in information retrieval (Simple Matching, Russel & Rao measure, Tanamoto Coefficient, Sorensen measure, Jaccard's Similarity [11]). Due to the sparseness of the binary matrix, the Jaccard similarity metric, which only considers the positive values in both the attribute value representations[7] , is used in this paper and set as default.

Concerning the second task, two new matrices express the two different affinity measures calculated by applying the selected similarity metrics on MT and MTR. The

[7] Let us define B_{11} as the total number of times a bit is ON in both bit strings, B_{00} as the total number of times a bit is OFF in both bit strings, and L as the length of the bit string, the Jaccard metric is defined as $B_{11}/(L - B_{00})$

matrices are built as follows. Given a matrix $AM(AMR)$ a generic element $e_{i,j}$ is obtained computing the similarity between the e_i and e_j rows of the matrix $MT(MTR)$, on the basis of the selected metrics.

Finally AM and AMR are linearly combined into the Global Affinity Matrix $GAM = \|gam_{hk}\|$. An element $gam_{hk} = lc_y \times am_{hk} + lc_m \times amr_{hk}$, where the values of lc_y and lc_m are chosen by the designer such that $lc_y, lc_m \in [0,1]$ and $lc_y + lc_m = 1$.

3.3 Step 3: Clustering technique selection

The prototype implements two different clustering algorithms: a classical agglomerative hierarchical clustering algorithm performs a partition of the values set, a second algorithm generates overlapping clusters (a variation of the algorithm in [5] is implemented).

The hierarchical clustering algorithm. A hierarchical clustering algorithm classifies elements into groups at different levels of affinity, forming a tree [6]. The hierarchical clustering procedure is applied to the matrix GAM. Once the affinity tree has been built, clusters are interactively computed on the basis of the numerical affinity values in the affinity tree and a threshold-based mechanism for cluster selection specified by the designer. High values of threshold return small, highly fragmented clusters. By decreasing the threshold value, bigger clusters are generated.

The overlapping clustering algorithm. The algorithm is based on the technique described in [5] and it is based on the idea of extending some sets of values given as input with other data set elements. In particular, the algorithm starts from a set of *poles* $\mathcal{P} = \{P_1, ..., P_l\}$ where P_i is a subset of the considered values set and $P_i \cap P_j = \{\}\ \forall i \neq j$. Then, a membership degree is calculated for each elements of the values set with respect to each pole. Finally, by means of a specific similarity measure evaluating the membership degrees, each element is assigned to one or more poles similar to it.

It is trivial to show that the results are highly dependent on the heuristic used for calculating the initial set of poles. Using the similarities available in our specific model, we implemented two techniques for calculating poles: the first one considers the results of the hierarchical clustering as poles, the second one considers the root elements as poles. The results are different: in the first case the similarity measures assume a key role; in the second case, no similarity measure is computed since the algorithm exploits only the domination.

3.4 Step 4: Name selection

A relevant value name is typically the most general value among the *values*, i.e. given a generic $rv_i = \langle rvn_i, values_i \rangle$, rvn_i is the most general value of $values_i$. The simplest way to detect a list of rvn_i candidates is to use the *Contains* function. The designer may select the most appropriate name among them.

3.5 Step 5: Validation

The results of clustering algorithms must be assessed with quality measures. We implemented a set of standard quality measures to support the designer in the tuning activity.

As stated in [7], two main cluster validation methods could be defined: external criteria, which are based on a comparison with a pre-specified cluster structure, provided by external knowledge (in our case human domain experts), and internal criteria, which are based on quantities that involve the vectors of the data set themselves. The measures we consider are the following:

- **countRV**: number of relevant values obtained for the configuration;
- **average, max_elements, variance**: the descriptive statistics over the number of elements; in particular, average expresses the average number of values belonging to a relevant value, max_elements indicates the dimension of the largest cluster and the variance shows the variance degree among the dimensions of the clusters; for values sets equally distributed on the domain max_elements is close to the average value and variance is low; the average value also gives an idea of the effectiveness of metadata representativity and "compression", all the elements included in non-outlier cluster are represented by the associated relevant value;
- **percentage of outliers**: best results are when it is close to that of reference set;
- **Rand Statistic index, Jaccard index, Folkes and Mallows index [7]**: compute the closeness of two sets of clusters evaluating couples of values that belong to the same cluster in both the sets;
- **silhouette [10]** (only if a hierarchical clustering algorithm is used): calculates for each object a silhouette value, which ranges from -1 (badly clustered) to 1 (well clustered); then, for each cluster calculates the average index; the global index in the table is the weighted average over all the clusters, excluding outliers;
- **overlapping degree** (only if an overlapping clustering algorithm is used): indicates the percentage of elements which belong to more than one relevant value.

Notice that the Rand, Jaccard, Folkes and Mallows indexes compare two different sets of clusters. We use these indexes both to compare the *RELEVANT* results w.r.t. a reference set, and to compare the differences between two different parameter settings. The reference set is provided by a domain expert or is a "gold standard".

4 Querying with Relevant Values

Thanks to the knowledge provided by relevant values, the user has two new ways of formulating queries, according to two scenarios.

1. The user has only a general idea of what she/he is searching for and composes a query predicate for instance by selecting a value *x* among the relevant value names. Note that instead of using the classical equality or LIKE operator, we should consider a new one, say RELATED TO, taking into account the mapping between relevant value names and values. It is beyond the scope of this paper to discuss such operator, but a naive implementation could be to substitute *At* RELATED TO *x* where *x* is a relevant value name, with *At* IN (SELECT *values* FROM METADATA.At WHERE *rvn*='x')
   To give a flavor of the novelty of the approach, we should observe that: (a) The user seldom has a deep knowledge of all the integrated data, so the list of the relevant value names, elicited from data, is of great help in providing insight on the

value domain, and in assisting query formulation; (b) w.r.t. the base SQL predicate At LIKE $'\%x\%'$ we propose a rewriting of the query which is guided by the semantics of clustering and string containment, and uses also, as base tools, the information retrieval techniques of stemming and stop words.

2. The user knows that the result must include tuples satisfying the predicate $At = v$, but she/he is aware that, due to the integration process, tuples with values v' *similar to* v might also be relevant. In this case the query could be transformed in a query of type 1 above by substituting $At = v$ with At RELATED TO rvn, where $v \in values(rvn)$, or possibly with a disjunction of predicates like that, if overlapping clustering is used.

5 Conclusions and future work

In this paper we defined a new type of metadata, the relevant values of an attribute domain. The experimental results evaluated by means of *RELEVANT* show that the technique produce results close to the relevant values provided by a domain expert. The best results are obtained by applying the overlapping clustering algorithms.

Future work will be addressed on improving the relevant values selection by automatically calculating some indicators for evaluating the quality of the relevant values. In this way, the designer may be supported in the parameters selection. Moreover, we will study the problem of the generation of the relevant value set for multiple attributes and that of quantitative evaluation of cluster quality in the overlapping case.

References

1. D. Beneventano and S. Bergamaschi. Semantic Search Engines based on Data Integration Systems. In *Semantic Web: Theory, Tools and Applications (Ed. Jorge Cardoso)*. Idea Group Publishing, 2006.
2. D. Beneventano, S. Bergamaschi, S. Bruschi, F. Guerra, M. Orsini, and M. Vincini. Instances navigation for querying integrated data from web-sites. In *In Int. Conf. on Web Information Systems and Technologies*, Setubal, Portugal, April 2006.
3. D. Beneventano, S. Bergamaschi, F. Guerra, and M. Vincini. Synthesizing an integrated ontology. *IEEE Internet Computing*, pages 42–51, Sep-Oct 2003.
4. S. Bergamaschi, S. Castano, D. Beneventano, and M. Vincini. Semantic integration of heterogeneous information sources. *Data & Knowledge Engineering*, 36(1):215–249, 2001.
5. G. Cleuziou, L. Martin, and C. Vrain. PoBOC: An overlapping clustering algorithm, application to rule-based classification and textual data. In *Proceedings of the 16th ECAI conference*, pages 440–444, 2004.
6. B. S. Everitt. *Cluster Analysis.* Edward Arnold and Halsted Press, 1993.
7. M. Halkidi, Y. Batistakis, and M. Vazirgiannis. On clustering validation techniques. *J. Intell. Inf. Syst.*, 17(2-3):107–145, 2001.
8. A. K. Jain, M. N. Murty, and P. J. Flynn. Data clustering: A review. *ACM Comput. Surv.*, 31(3):264–323, 1999.
9. M. Lenzerini. Data integration: A theoretical perspective. In Lucian Popa, editor, *PODS*, pages 233–246. ACM, 2002.
10. P.J. Rousseeuw. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *J. Comput. Appl. Math.*, 20:53–65, 1987.
11. C. J. van Rijsbergen. *Information Retrieval.* Butterworth, 1979.