RELEVANT VALUES: NEW METADATA TO PROVIDE INSIGHT ON ATTRIBUTE VALUES AT SCHEMA LEVEL

Sonia Bergamaschi, Mirko Orsini
Dipartimento di Ingegneria dell'Informazione, Università di Modena e Reggio Emilia, Italy
firstname.lastname@unimore.it

Francesco Guerra
Dipartimento di Economia Aziendale, Università di Modena e Reggio Emilia, Italy
francesco.guerra@unimore.it

Claudio Sartori
Dipartimento di Elettronica, Informatica e Sistemistica, Università di Bologna, Italy
claudio.sartori@unibo.it

Keywords: Metadata, querying, data integration, data mining

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, as in the Semantic Web context.

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 (Lenzerini, 2002; Beneventano and Bergamaschi,) 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 extensional knowledge emerging from the attribute values. We call these metadata "relevant values" as they provide the users 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 when 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 thus significantly limiting 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 attribute value, 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 gener-

ate 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[1] (Bergamaschi et al., 2001), 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 (Beneventano et al., 2003), 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[2], 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 (Beneventano et al., 2006) a first heuristic for calculating relevant values was described.

In this paper we improve the approach proposed in (Beneventano et al., 2006), 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 (Jain et al., 1999) 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.

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

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

2 Eliciting Relevant Values from Data

In the context of the data integration, there are several models for representing ontologies. Without loss of generality, let us refer to the concepts of MOMIS. The GVV built with MOMIS is composed of Global Classes (i.e. OWL Classes), with Global Attributes (GA), i.e. OWL datatype attributes . 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.

Definition 1 *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, while* $values^{At}$ *is the set of values referring to it, i.e.* $values^{At} \subseteq \pi_{At}(C)$.

The set of the relevant values of At, $RVS = \{rv_1, rv_2, \ldots, rv_n\}$ has two attributes : 1) $\bigcup_{i=1}^{n} values_i = \pi_{At}(C)$ and 2) $rvn_i \neq rvn_j \forall i \neq j$. For simplicity, in the following we will drop the At index.

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 data mining clustering techniques, adapted to the problem on hand; the second will require the intervention of the integration designer, but we will provide an effective *assistant*.

2.1 Adding semantics: the root elements

Nomina sunt consequentia rerum
Giustiniano, Institutiones, Liber II, 7, 3 [3]

On the basis of an analysis of manufacturing databases, we observed that it is frequent to have string domains with values composed by many words, also with abbreviations. We observed also that the same word, or group of words, may be further qualified 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, it is possible to devise a lattice of similar values. The lattice semantics has to be verified by the integration designer for the specific attribute domain, but, in our experience, it is sound.

Our idea is to exploit the semantics expressed by such containment for building clusters of values. The

[3] Names are consequences, or, one might say, the expressions, of things.

proposed technique finds out the "relevant" words included in attribute values and uses them as roots to build a lattice over the flat set of values. The lattice will be used together with the usual syntactic clustering methods, in order to effectively group values.

Definition 2 *A **root element** $re = \langle ren, values \rangle$ is a relevant value computed by means of a function based on string containment: $Contains(X,Y) = true$ iff $stem(X) \supseteq stem(Y)$, where X and Y are sets of words.*

A root element name ren_j is defined as a value of the domain such that it does not contain any other value of the domain, formally:

$ren \in \pi_{At}(C) : \nexists x \in \pi_{At}(C), Contains(ren,x), ren \neq x.$

The values of a root element are the values of the domain containing the root element name, i.e. $values = \{v \in \pi_{At}(C) : Contains(v,ren)\}$.

The definition implies that the set of all the root elements is closed w.r.t. the domain, i.e., if there exists m root elements, $\bigcup_{j=1}^{m} values_j = \pi_{At}(C)$. RES is the set of all the root elements of a property.

Our claim, supported by experience, is that root elements do bear semantics, since they derive directly from the linguistic choices of people who populated the data source.

2.2 Relevant Value Sets

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, by mapping each value into the universe of words generated considering all the values of the property.

At present, we implemented two ways to build some structure upon the flat set binary representation: 1) the semantics of *containment*, expressed by the root elements defined above; and 2) the semantics of *clusters*, mapping the words of attribute values in an abstract space, and defining a syntactic similarity function in such space.

The similarity measures computed with both the semantics are then used by a clustering algorithm (in *RELEVANT* the user may generate both partitions and overlapped clusters). The algorithm produces clusters of *values* which are the Relevant Values set.

2.3 Relevant Value Names

The name of a relevant value *rvn* has to be representative of the elements belonging to the *values* set associated. Consequently, let us consider the set of relevant values RVS associated to a specific attribute At. The set of $RVN = \{rvn_1, ..., rvn_n\}$ provides the integration designer with a synthesized knowledge about the At contents.

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 integration designer may select the most appropriate among them.

In particular, let us consider a $rv_i \in RVS$. The list of rvn_i candidates is calculated as follows: let $\bar{s} = \{v \in values_i \mid Contains(v',v) = true\ ,\ v \neq v', and\ \nexists v'' \in values_i \mid Contains(v,v'') = true\ v \neq v''\}$. If $\bar{s}$ is a singleton, then $rvn_i = \bar{s}$, otherwise the integration designer has to select the name among the elements of $\bar{s}$, or introduce another name.

3 The *RELEVANT* prototype

Figure 1 shows the *RELEVANT* functional architecture, which is organized into four blocks:

1. **Data pre-processing**: a binary representation of the values of an attribute is obtained with a matrix where the columns represent the universe of words (the extension of the property) and each row represents the mapping of an attribute value.
2. **Similarity Computation**: two tasks are enabled: the selection of the metrics for computing the similarity between pairs of attribute values and the selection of the semantics to be used: the semantics of containment, the semantics of clusters or a combination of both.
3. **Relevant Values elicitation**: this module implements some clustering algorithms to compute the set of relevant values on the basis of the choices made at step 2.
4. **Validation**: we implemented some standard techniques to evaluate cluster quality. At present, additional work and experiments are necessary go further the simple evaluation, so as to provide effective assistance to the designer in the critical task of parameter configuration.

3.1 Step 1: Binary Representation of attribute values

The starting point of the process is the *Matching Table* MT, that is a binary representation of the values of a property, obtained as follows:

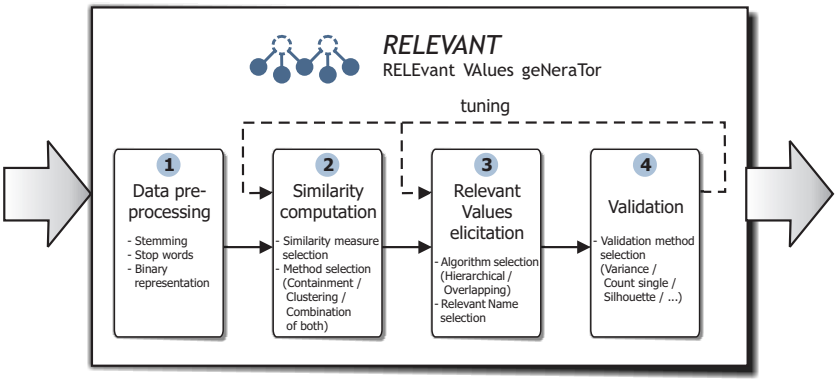

Figure 1: The *RELEVANT* functional architecture

1. Let $W = \pi_{At}(C) = \{w^i\}$, the extension of At. Each string w^i is found in at least one value and is a set of elementary words, say $w^i = \{w^i_k\}$.
2. From *information retrieval* we import the concept of *stemming*, and we use a *stemming operator* for words, such that $stem(w) = a$, where a is either a stem form of w or $\emptyset$, if w is a *stop–word*. For string and sets of strings the definition of stem() is easily extended: $stem(s) = \{stem(w_i)\}$ for $s = \{w_i\}$ and $stem(S) = \{stem(s^i)\}$ for $S = \{s^i\}$.
3. The universe of words considered is then
$$U = \bigcup_{w^i \in W} stem(w^i) = \{a_k\},$$
where a_k are numbered in lexicographic order.
4. The binary representation of At is described by means of the rows of MT, where
$$MT = \|mt_{xy}\|_{|W|\times|U|}, m_{xy} = \begin{cases} 1 \text{ if } a_y \in stem(w^x) \\ 0 \text{ otherwise} \end{cases}$$

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 property, except for the stop–words. Our approach applies clustering techniques to MT to obtain sets of similar values and gather them in a single relevant value.

3.2 Step 2: Similarity computation

The binary representation is used to calculate similarity between attribute values. The computed similarity is then exploited by a clustering algorithm to create sets of *values*. Two tasks are executed in this step: the similarity measure and the selection of the kind of "exploited" semantics.

Concerning the first task, we consider some well-known similarity measures collected in (Luke,) (e.g.: Simple matching, Jaccard,...). The choice of the similarity measure is orthogonal w.r.t. the clustering algorithm, and *RELEVANT* sees it as a parameter, which can be chosen by the integration designer and changed to compare different settings. We build the "Affinity Matrix" $AM = \|am_{hk}\|_{|W|\times|W|}$ where am_{hk} is the similarity between the mt_h and mt_k rows of the matrix MT, binary representation of the original values a^h and $a^k \in At$.

Concerning the second task, it is trivial to show that syntactically similar values may refer to very different objects: consequently the use of the semantics of clusters may produce improper clusters. On the other hand, adding semantics to each attribute value is a time-consuming and error-prone operation: the high number of values and the frequency of update discourage a manual processing as required by the usual techniques (e.g.: by annotating each value according to a reference ontology).

Our choice is to introduce a new technique that grasps the semantics of the values containment (expressed by the Root Elements) working in conjunction with the approach based on the above defined semantics of cluster which exploits the syntactic similarity. The designer may balance the relevance of the two kind of semantics by means of specific coefficients.

To exploit such semantics, we introduce the Matching Table for Root elements MTR, a binary matrix that shows the membership of the attribute values to the Root Elements. In particular, $MTR = \|mtr_{hk}\|_{|W|\times|SRE|}$, where $mtr_{hk} = \begin{cases} 1 \text{ if } a^h \in values_k \\ 0 \text{ otherwise} \end{cases}$

Therefore, a new matrix, Affinity Matrix for Root elements (AMR), is built.

$AMR = \|amr_{xy}\|_{|W|\times|W|}$ represents the similarity measure between two elements mtr_x and mtr_y. This measure is computed by means of the metric selected by the integration designer.

Finally the results of *AM* and *AMR* are linearly combined into the Global Affinity Matrix $GAM = \|gam_{hk}\|_{|W|\times|W|}$. In particular, $gam_{hk} = lc_y \times am_{hk} + lc_m \times amr_{hk}$, 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: Relevant values elicitation

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 (Cleuziou et al., 2004) is implemented).

The hierarchical clustering algorithm. A hierarchical clustering algorithm classifies elements into groups at different levels of affinity, forming a tree (Everitt, 1993). 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 (Cleuziou et al., 2004) 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 semantics 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, as shown below. 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 containment property.

3.4 Step 4: Validation

We implemented a set of standard quality measures, to support the designer in the tuning activity. In particular, the tool provides:

- **countRV**: number of relevant values obtained for the configuration. This value depends on the threshold set by the integration designer;
- **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 set equally distributed on the domain max_elements is close to the average value and variance is low;
- **count single**: number of relevant values with a single element. Count single indicates a low number if the values set is equally distributed on the domain;
- **silhouette (Rousseeuw, 1987)** (only if the hierarchical clustering algorithm is used): calculates a width for each cluster based on the comparison of its tightness and separation. If the silhouette value is close to 1, it means that the object is *well-clustered* and has been assigned to a appropriate cluster. If the silhouette value is close to `-1`, it means that the object is *not well-clustered*;
- **overlapping degree** (only if the overlapping clustering algorithm is used): indicates number of elements which are in more than one relevant value.

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 . These values are provided to the user in order to increase his sources knowledge. We addressed several critical issues with the aim of efficiency and effectiveness, in different domains with different updating frequencies. In particular, the method is based on data analysis: if data change, the relevant values have to be updated. As usual in data analysis, the startup phase requires the setting of several critical parameters. Nevertheless, for a given parameter setting, the developed technique is able to calculate the relevant value set without any human intervention. Moreover the parameters and similarity metric selection determine the quality of the relevant value set. Therefore, the integration designer has to carefully evaluate the results and eventually change some parameter setting in order to improve the quality result.

The experimental results evaluated by means of *RELEVANT* show that the developed technique may 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.

Acknowledgments

This work was partially supported by the Italian Ministry of University and Research within the projects WISDOM and NeP4B.

REFERENCES

Beneventano, D. and Bergamaschi, S. Semantic Search Engines based on Data Integration Systems. In *In Semantic Web: Theory, Tools and Applications (Ed. Jorge Cardoso)*. Idea Group Publishing, forthcoming (see http://www.dbgroup.unimo.it).

Beneventano, D., Bergamaschi, S., Bruschi, S., Guerra, F., Orsini, M., and Vincini, M. (2006). Instances navigation for querying integrated data from web-sites. In *In International Conference on Web Information Systems and Technologies (WEBIST 2006)*, Setubal, Portugal.

Beneventano, D., Bergamaschi, S., Guerra, F., and Vincini, M. (2003). Synthesizing an integrated ontology. *IEEE Internet Computing*, pages 42–51.

Bergamaschi, S., Castano, S., Beneventano, D., and Vincini, M. (2001). Semantic integration of heterogeneous information sources. *Data & Knowledge Engineering, Special Issue on Intelligent Information Integration*, 36(1):215–249.

Cleuziou, G., Martin, L., and Vrain, C. (2004). PoBOC: An overlapping clustering algorithm, application to rule-based classification and textual data. In *Proceedings of the 16th ECAI conference*, pages 440–444.

Everitt, B. S. (1993). *Cluster Analysis.* Edward Arnold and Halsted Press.

Jain, A. K., Murty, M. N., and Flynn, P. J. (1999). Data clustering: A review. *ACM Comput. Surv.*, 31(3):264–323.

Lenzerini, M. (2002). Data integration: A theoretical perspective. In Popa, L., editor, *PODS*, pages 233–246. ACM.

Luke, B. T. Clustering binary objects. In *Technical Report*, http://fconyx.ncifcrf.gov/ lukeb/binclus.html.

Rousseeuw, P. (1987). Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *J. Comput. Appl. Math.*, 20:53–65.