Data Structure Estimation for RDF Oriented Repository Building (Extended Abstract)[1)]

Martin Řimnáč
Institute of Computer Science, Czech Republic, Europe.
Email: rimnacm@cs.cas.cz

Abstract:
Mechanisms for accessing and training the data repository using a binary matrix formalism are presented. The repository is designed for a data storage through corresponding instances of simple attribute functional dependencies, which can be seen as similar to the binary predicate formalism being used by the RDF semantic web format.
Two mechanisms for querying a repository, the generalisation and the specialisation, are given. Furthermore, the incremental repository training mechanism with no extra requirements on the input data form is described: The extensional functional dependency system is used as a generalised view on the stored data; the algorithm is inspired by the functional dependency discovery approach.

1 Introduction

The aim of the paper is to propose a system, which enables unknown structured data to be stored and consequently to be accessed through search queries; these queries are based on the specialisation (which objects satisfy a condition given by a query) and the generalisation (which fact are valid for all objects satisfying a query). The generalised relationships generated by the features of instances in the repository will be found very important for the internal organisation; a generalised description by a functional dependency system for a data structure is chosen.

The proposed system uses the binary matrices for the internal representation and is restricted to relationships between single attributes (similar to the RDF semantic web format [1, 2]). The proposal is also inspired by results of the functional dependency discovery approach [3, 4, 5, 6, 7], which estimates a data structure by an extensional functional dependency system.

[1)]The work was supported by the project 1ET100300419 of the Program Information Society (of the Thematic Program II of the National Research Program of the Czech Republic) "Intelligent Models, Algorithms, Methods and Tools for the Semantic Web Realization" and partly by the Institutional Research Plan AV0Z10300504 "Computer Science for the Information Society: Models, Algorithms, Applications".

Note, the returned model $M(R')$ always principally depends on its input relation R' and can be seen only as an estimation and may be changed upon the relation R' will be extended by new tuples. The extensional model can be generated using the extensional dependency definition

$$(A_i \rightarrow_{R'} A_j) \in M_{R'} \Leftrightarrow \exists \mathscr{P}' : \mathscr{D}_\alpha^{R'}(A_i) \rightarrow \mathscr{D}_\alpha^{R'}(A_j) \tag{1}$$

2 Repository Definition and Retrieval

The repository is a system enabling data storage and retrieval. Let a binary matrix notion be used for the internal representation: Denote by $\mathscr{T}$ a set of tuples representing a part of universum. Further, let $\mathscr{A}$ be the set of attributes corresponding to the set of tuples $\mathscr{T}$, $t_i(A)$ a value of the attribute $A \in \mathscr{A}$ in the tuple $t_i \in \mathscr{T}$ and $\mathscr{D}$ a set of all values. The set $\mathscr{A}$, respectively $\mathscr{D}$ is also accessible thought the index $I_\mathscr{A}$, respectively $I_\mathscr{D}$ given by the mapping $\mathscr{A} \rightarrow N$, respectively $\mathscr{D} \rightarrow N$. The basic element in the repository will be then an attribute-value pair, which will be indexed by the corresponding projection $I_\mathscr{E} : \mathscr{A} \times \mathscr{D} \rightarrow N$.

The `active domain matrix` is a binary matrix $\Delta_\mathscr{T} = [\delta_{ij}]$ indicating for each element (A, v) with the index $i = I_\mathscr{E}(A, v)$ and each attribute $A \in \mathscr{A}$ with the index $j = I_\mathscr{A}(A)$ the fact $v \in \mathscr{D}_\alpha(A)$:

$$\Delta_\mathscr{T} = \{\delta_{ij}\} : \delta_{ij} = \begin{cases} 1 & \text{if } \exists t \in \mathscr{T} : i = I_\mathscr{E}(A, t(A)), j = I_\mathscr{A}(A) \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

The `extensional functional dependency matrix` $\Omega_\mathscr{T} = \{\omega_{ij}\}$ is a binary square matrix representing a projection $I_\mathscr{A} \times I_\mathscr{A} \rightarrow \{0, 1\}$:

$$\Omega_\mathscr{T} = \{\omega_{ij}\} : \omega_{ij} = \begin{cases} 1 & \text{if } (A_j \rightarrow A_i) \in M_\mathscr{T} \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

Analogically, the `instances` $\Phi_\mathscr{T} = \{\phi_{ij}\}$ is organised as a binary square matrix, where

$$\Phi_\mathscr{T} = \{\phi_{ij}\} : \phi_{ij} = \begin{cases} 1 & \text{if } j = I_\mathscr{E}(A_1, v_1) \text{ implies } i = I_\mathscr{E}(A_2, v_2), \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

The instance matrix $\Phi_\mathscr{T}$ generates the general description of the relationships – the extensional functional dependency matrix $\Omega_\mathscr{T}$ – using

$$\Omega_\mathscr{T} = \Delta_\mathscr{T}^T \cdot \Phi_\mathscr{T} \cdot \Delta_\mathscr{T} \tag{5}$$

Further in the inverse way, the functional dependency matrix $\Omega_\mathscr{T}$ implies the mask of positions in the instance matrix $\Phi_\mathscr{T}^M$, which can be set; the rest of positions can not be activated due to the fact that there is no corresponding functional dependency in the $\Omega_\mathscr{T}$ matrix.

$$\Phi_\mathscr{T}^M = \Delta_\mathscr{T} \cdot \Omega_\mathscr{T} \cdot \Delta_\mathscr{T}^T \tag{6}$$

2.1 Repository Searching

Let $\vec{x_0}$ be a vector representing elements to be searched for. There are two querying mechanisms describing one step:

- The `generalisation` – the vector $\vec{x}_k$ is extended by element(s), which are implied by any element in the vector.

$$\vec{x}^G_{k+1} = \Phi_{\mathscr{T}} \cdot \vec{x}_k \tag{7}$$

- The `specialisation` – the result gives elements, which can imply any element(s) in the vector. Note, the result may correspond to several different objects (and if required, these objects have to be separated after each step is done).

$$\vec{x}^S_{k+1} = \Phi^T_{\mathscr{T}} \cdot \vec{x}_k \tag{8}$$

On the condition each tuple $t_i \in \mathscr{T}$ contains the key attribute A_k, only instances of functional dependencies between single attributes can be considered. In such a case, complex attribute relationships can be expressed as extra-added meta-information, the generalisation is extended by (γ_L, γ_R are the left/right sides of the functional dependency)[1)]:

$$\vec{x}'_{k+1} = \Phi \cdot \vec{x}_k + \Phi((\Phi^T \cdot ((\Delta^T \cdot \gamma_L) \odot \vec{x}_k)) == 1) \odot (\Delta^T \cdot \gamma_R) \tag{9}$$

2.2 Repository Matrix Forms

The matrix $\Phi_{\mathscr{T}}$, respective $\Omega_{\mathscr{T}}$ can be expressed in one of these forms:

- `Minimal`: This form does not contain any redundant items (optimal for an effective repository storage), but the result can be reached generally in several steps (κ^G/κ^S for the $\Phi_{\mathscr{T}}$ generalisation/specialisation, when Φ reasoning is monotonic):

$$1 \leq \kappa^G \leq |\mathscr{A}| \tag{10}$$

$$1 \leq \kappa^S \leq |I_{\mathscr{E}}| \tag{11}$$

- `Full`: The matrix may contain a lot of redundant items, but the result is reachable in one step (optimal for the reasoning).

The minimal form of the instance matrix $\Phi'_{\mathscr{T}}$ for an effective repository storage can be handled by the incremental model skeleton $\Omega'_{\mathscr{T}}$ finding algorithm [8] followed by the corresponding position mask (6) restriction.

[1)] The operator $\odot$ definition: $C = A \odot B$ when $\forall i, j : c_{ij} = a_{ij} b_{ij}$

3 Repository Training from Input Data Set

This section describes a way the repository including the instance matrix $\Phi_{\mathscr{T}}$ and the functional dependency matrix $\Omega_{\mathscr{T}}$ can be created from the set of tuples $\mathscr{T}$. The algorithm stores (in k-th step) the instance matrix Φ_k and the corrupted functional dependency matrix $\mho_k$; the functional dependency matrix Ω_k can be derived by (5) and the active domain matrix Δ is known[2)].

The incremental algorithm is based on the fact that the functional dependencies and their instances can be easily generated for a model covering only one tuple (in such a case, each attribute value can be derived for any other attribute value) - the tuple corresponds to the square block of 1.

The algorithm input is a set of matrices $\{\Phi_i^{\Delta} = \vec{x_i}^T \cdot \vec{x_i}\}\ , \vec{x_i} \in \mathscr{T}$, the output is represented by the corrupted functional dependency matrix $\mho_{\mathscr{T}}$ and the instance matrix $\Phi_{\mathscr{T}}$. The algorithm without determining the complex attribute functional dependencies has the following structure:

- Merging the current state instance matrix Φ_k and the k-th input tuple matrix Φ_k^{Δ}:

$$\Phi'_{k+1} = \Phi_k + (\Delta^T \cdot (1 - \mho_k) \cdot \Delta) \odot \Phi_k^{\Delta} \tag{12}$$

- Detecting corrupted functional dependencies (by inconsistent items implying more than one element per attribute from $\vec{x}_k$:

$$\mho_{k+1}^{\Delta} = \Delta((\Phi_{k+1}^T \cdot \Delta^T) \gg 1) \tag{13}$$

- Removing the inconsistent items by (6):

$$\Phi_{k+1} = \Phi'_{k+1} \odot (\Delta^T \cdot (1 - \mho_{k+1}^{\Delta}) \cdot \Delta) \tag{14}$$

- Actualising the corrupted functional dependency matrix:

$$\mho_{k+1} = \mho_k + \mho_k^{\Delta} \tag{15}$$

- Continue with next k while all input matrices are not gathered. Finally:

$$\mho_{\mathscr{T}} = \mho_k,\ \Phi_{\mathscr{T}} = \Phi_k \tag{16}$$

[2)] In this proposal, Δ is designed as static, in the real algorithm, the matrix has to be updated before new tuple gathering.

4 Conclusion

The paper dealt with a binary matrix formalism usage for handling the repository described by relationship meta-description and instances expressing the binary predicates, i.e. to the relationships between single attributes. The main paper idea is to make the introduction at first to the basic mechanisms to operate in the repository (specialisation and generalisation mechanisms) and secondary to describe a way the input data $\mathscr{T}$ can be stored in the repository.

The consideration of a structure for a global relationship description by the extensional functional dependency system was found useful; The relationships are driven by input data and reversibly affect the repository during next tuple insertion.

Finally, the repository can be seen as triples - a RDF semantic web document, because the same expression formalism - binary predicates - is used. In this situation, the data structure defines/estimates the semantics in an explicit way.

Generally, the proposed mechanisms can be used for storing and querying data from the repository. This incremental proposal being useful for handling data with an a-priori unknown structure is especially oriented towards to gathering data from web page content extractors.

The future work aim is to precise the complexity issue by consideration of sparse matrices and to extend and to test this algorithm for tree formated documents (XHTML).

References

[1] G. Antoniou, F. v Harmelen. "A Semantic Web Primer".
MIT Press, 2004. ISBN: 0-262-01210-3.

[2] E. Miller, R. Swick, D. Brickley. "Resource Description Framework".
`<http://www.w3.org/RDF/>` [on-line]. 2004.

[3] H. Mannila, K.J. Räithä. "Design by Example: An Applications of Armstrong Relations."
Journal of computer and system sciences 33, pp. 129-141. Academic Press. 1986.

[4] H. Mannila, K.J. Räihä "Dependency Inference".
In *Proc. of VLDB*. pp. 155–158. ISBN: 0-934613-46-X. 1987.

[5] H. Mannila, K.J. Räithä. "Algorithms for Inferring Functional Dependencies from Relations." In *Data & Knowledge Engineering* 12, pp 83-99. Elsevier. 1994.

[6] J. Kivinen, H. Mannila "Approximate Inference of Functional Depentencies from Relations". In *Proc. of 4. int. conf. on Database Theory*, Berlin, Germany. pp. 129–149. ISSN: 0304-3975. 1995.

[7] P.A.Flach, I.Savnik. "Database Dependency Discovery: A Machine Learnig Approach".
In *AI Communications*, Volume 12/3, pp. 139–160. 1999.

[8] M. Rimnac "Transforming Current Web Sources for Semantic Web Usage" In *Proc. of SOFSEM 2006. Volume 2. pp 155-165 (ISBN: 80-903298-4-5). 2006.*