An integrated data mining and data presentation tool

F. Angiulli 4, T. Catarci1, P. Ciaccia2, G. Ianni3, S. Kimani1, S. Lodi2, M. Patella2, G. Santucci1, C. Sartori2

1Dipartimento di Informatica e Sistemistica, Università di Roma, Italy
2Dipartimento di Elettronica, Informatica e Sistemistica, Università di Bologna, Italy
3Dipartimento di Elettronica, Informatica e Sistemistica, Università della Calabria, Italy
4ISI - CNR, DEIS, Università della Calabria, Italy

Abstract

The process of data mining includes many steps, starting with the choice and preparation of data sources and ending with the presentation of the data mining results. In addition, it is generally accepted that the data mining is not a “one shot” process, but rather the result is obtained through iterative refinement steps of algorithm choice, parameters settings and intermediate results presentation. An effective architecture for a data mining tool should therefore allow easy integration of three components: acquisition of data sources, data mining algorithms and results presentation. In this paper we present the architecture of a data mining tool which is under development in the framework of the project D2I (Data to Information), supported by the Italian MIUR (Ministry of Instruction, University and Research). The architecture is based on the concept of “metadata repository”: it is a specification of the data exchanged by the various modules which guarantees flexibility and extensibility: new algorithm and presentation method can be added, provided that the metadata specification is available. As a guideline and a testbed for the architecture, we present the specification of some data mining methods and we sketch how their results can be presented.
1 Introduction

The knowledge discovery process entails more than just the application of data mining strategies. There are many other aspects including, but not limited to: planning, data pre-processing, data integration, evaluation and presentation. In fact, the discovery process is both a domain-centered process and a human-centered process. On the whole, there is a need for an overall framework that can support the entire discovery process. Of special interest, is the role and place of visualization in such a framework. Visualization enables or triggers the user to use his/her outstanding visual and mental capabilities, thereby gaining insight and understanding of data. In the following we point out the pivotal role that visualization can play in supporting the user throughout the entire knowledge discovery process, whereas, traditionally, visualization has been placed at the beginning and at the end of the knowledge discovery process.

A key feature of a knowledge discovery system based on visualization should be its extensibility on two orthogonal dimensions: i) adding new paradigms for visualization and making them available for the widest range of data mining results, and ii) adding new data mining algorithms and making their results presentable with the widest range of visualization techniques. For this reason we devised an architecture based on the following guidelines:

- the system must be thought to present the user an heterogeneous set of tasks in the most homogeneous and integrated way;
- the system must be multi–platform;
- the system must be open;
- the system must have a modular structure with well–defined change/extension points;
- the system must provide the end user with the maximum flexibility during his data mining tasks.

The main problem lies in the support of heterogeneous data mining methods: this affects both the internal system behaviour (e.g. algorithm activation, results exchange) and the user interface (e.g. goal description, result presentation). To overcome these difficulties, a deep abstraction effort has been necessary in the development of the architecture. Beside this, some architectural guidelines has been followed:

- the system is internally multi layer;
- specific extension points have been defined in the architecture where the additions or changes of data mining user interactions, procedures and algorithms concentrate.

The overall architecture of the tool is shown in Figure 1 and presents the following main layers:

1. the graphical user interface, intended for handling the user interaction in all data mining activities, from data definition to result understanding;
2. the data mining engines, i.e. the implementations of the data mining processes;
3. the metadata support, intended to make easily accessible to the other modules all the information related to the data mining processes, which is fed by the data mining modules for the results and by the visualization module for the parameters of each data mining experiment, and makes these data available for every system module.

Communications between the modules is supported by an intersystem communication subsystem, providing a specific set of communication primitives able to manage all the possible interactions and data transfer of different data mining techniques in a uniform way.

In this paper we will describe in some detail the metadata repositories for the data mining activities which have been considered so far and our presentation choices. The structure of the paper is the following: in Section 2 we describe metadata support. Section 3 describes the User Interface and Visualization module, including our proposals for dealing with a particular kind of data exploration: the similarity queries. Finally, in Section 4 we conclude the paper.

2 Repository and interchange formats for metadata support

Metadata support in the system is implemented in two ways: (i) By a metadata repository, (ii) by the specification of interchange formats. The metadata repository records the relationships among the data sources, parameters, and outputs of data mining tasks. Interchange formats are specifications of XML documents
which enable data mining engines to exchange input/output information with the GUI.

**Metadata Repository** The structure of the data sources, the parameters, and the outputs of a data mining task can be designed at the meta-level by expressing suitable abstractions of the involved elements and constraints. Of course, such meta-representation can be naturally designed in many different ways. In order to enhance modularity and keep system architecture as open as possible, we define a generalization meta-schema, expressing the structure which can be shared by most data mining tasks.

The proposed generalization metaschema is depicted in Figure 2. Every source of data MiningSource and mining output MiningOutput is represented by a table object MiningClass which has at least one column MiningAttribute. Every output object is uniquely determined by the mining task which generated the output, the parameters used in the run, the source which provided the data to the task, and an identifier (for tasks generating multi-relational output).

Each data mining method which specializes the Abstract Data Mining Engine should extend the metaschema, by adding entities and relationships describing the fine details of the mining models computed by the method.

**Interchange Formats** The Predictive Model Markup Language (PMML) XML DTD has been proposed as a standard for exchanging mining results. Such proposal is extended by the metarules and clustering interchange formats, by allowing for (i) metaqueries (whereas PMML represents traditional association rules only), and (ii) the selection of a larger number of clustering methods and specification of input parameters.

In the following we review the mining tasks currently supported by the system, and describe the extension to the metadata repository provided by the clustering and approximate similarity queries engines, and the Metarules Interchange Format
Data Mining III

(MIF). We omit, due to lack of space, the repository extension provided by the metaquerying engine and the interchange formats specified for the clustering and approximate similarity queries engines. (The interested reader may refer to [1][3].)

2.1 Methods overview

Metaqueries [7] are generic descriptions of classes of patterns from a relational database. For instance, a metaquery \( T \leftarrow L_1, \ldots, L_m \), where \( T \) and \( L_i \) are literal schemes \( Q(Y_1, \ldots, Y_n) \), can be instantiated, or answered, by substituting the predicate variables with predicate (i.e. relation) names from the database. Notice that, unlike association rules, metaqueries link information from several tables. Therefore a typical application of metaquerying is in discovering conditional patterns in complex relational databases.

The goal of data clustering is to cover the data set, usually consisting of one relation, with maximally homogeneous and maximally separated subsets, where homogeneity is often made precise by means of a dissimilarity function on objects, having low values at pairs of objects in one cluster.

Similarity queries between two objects are frequently used in data exploration and mining, e.g. as a search routine in clustering algorithms, or in the iterative exploration (by query execution and refinement) of multimedia databases. In all applications (such as the above) where multiple expensive similarity queries have to be executed, the user may thus be willing to accept a time/quality trade-off, i.e. an error with respect to the exact case is traded for an often dramatic improvement in execution time. In its essence, the problem is to find objects which are similar, up to a given degree, to a given query object. Typical similarity queries include range queries (where all the objects in \( O \) whose distance to \( q \) does not exceed a user-specified threshold \( \alpha \) are requested) and \( k \)-nearest neighbor (k-NN) queries (where the \( k \) objects in \( O \) which are closest to \( q \) are requested).

Approximate evaluation of similarity queries requires the definition of an error measure [2, Part III]. The PAC approach differs from correct (where no error is allowed) and approximately correct (where the error cannot exceed a specified bound) search because it allows to exceed the error bound \( \varepsilon \) with a certain probability \( \delta \). The specification of PAC algorithms, therefore, relies on the possibility of defining a suitable measure of error and on finding a way to limit such error in a probabilistic way. In [4, Part III], we proposed error measures for both range and k-NN queries, and presented algorithms for solving PAC queries with M-tree [5].

The input for PAC queries is a point and three parameters, i.e. the values of \( \varepsilon \), \( \delta \), and \( \alpha \) or \( k \) depending on the query type (range or k-NN). The output is just a set of objects.

2.2 The data clustering repository

The ER metaschema (see Figure 3) for the data clustering task consists of classes matching the ones in the generalization metaschema as well as labeling information, and various kinds of summaries for each cluster.
Data to be clustered and results of the clustering task are represented by ClusteringDataSource and ClusteringOutput. For simplicity, here we use a simple attribute ClusteringFunction to represent succinctly the clustering function including every parameter needed. ClusteringOutput also represents the labeling relationship. For each clustering output, labeling assigns to each data object one or more cluster labels. (Every object is assigned exactly one cluster label if the clustering function outputs a partition rather than a cover of the data set.) The remaining parts of the metaschema deal with summary information. For each cluster, we consider the following information: For each attribute and value of the attribute, the relative frequency of the value, i.e. the fraction of objects with respect to all the objects in the cluster which support the value is stored in the summary. The entity Cluster represents a cluster of the output, uniquely identified within the output. ClusterFrequencies maps every attribute cluster, and attribute value to a frequency Frequency. For data sources, we use similarly DataSourceFrequencies as the class of relative frequencies of a value of an attribute in a data source.

2.3 The metarules interchange format

Assume a metaquery has to be evaluated. A document in MIFIn format is produced, and submitted for evaluation.

In general, each MIFIn document contains a <metaquery> tag, which contains two subelements, <head> and <body>. <head> and <body> may contain, respectively, one or many <metaatom> tags, which specify single literal patterns. A <metaatom> tag contains a list of variables denoted by the <variable> tag. Each tag carries some specific attribute. The <metaquery> tag has the attributes support and confidence, whose value expresses the required threshold level for support and confidence on instantiated answers. A <metaatom> tag may contain either a sequence of elements of the kind <fitRelation> or a sequence of elements of the kind <unfitRelation>. A sequence of <fitRe-
<relation> elements specifies a set of relation names to which the predicate variable at hand can be matched to. A sequence of <unfitRelation> elements specifies a set of relation names to which the predicate variable at hand cannot be matched to, instead. If none of the above sequences are present, a predicate variable may be freely matched to any relation. A <metaatom> contains anyway a list of <variable> tags, which specifies the set of ordinary variables associated to the metaatom at hand. The meaning of the elements <fitAttributeName> and <unfitAttributeName> contained within the <variable> element is symmetric to the elements <fitRelation> and <unfitRelation> of the <metaatom> tag. The element <fitDataType> (resp. <unfitDataType>) may be employed in order to specify which data type the variable at hand could be matched to (resp. not matched to). Note that it is possible to employ these statement within any occurrence of the tag <variable>, referring to the variable at hand.

The MIFOut format is simpler than MIFIn, since it is designed to transport sets of instantiated rules giving necessary details on how instantiations were performed. A MIFOut document contains a sequence of <rule> tags, one for each instantiated rule, with the associated values of support and confidence reported in the attributes support and confidence. Each tag <rule> contains the tags <head> and <body>. The former contains a single tag <atom>, while the latter contain a sequence of tags <atom>. The tag <atom> placed into the tag <head> specifies the head of the instantiated rule, while the tags <atom> placed into the tag <body> describe the body of the rule. A tag <atom> has an attribute name, whose value is the name of a relation of the data base, and contains a sequence of <variable> tags, with attributes name (the name of the variable) and attributeName (the name of the associated attribute).

Two XML Schemes are defined for the MIFIn and MIFOut format respectively. The two schemes are employed in order to make applications able to easily validate MIFIn and MIFOut formats, using standard parsing tools.

3 The user interface and visualisation

The system provides the user with a consistent, uniform and flexible interaction environment across the entire process of mining knowledge. The visual environment is intended to place the user at the center of the entire data mining process. The interface employs various visual strategies that can effectively enable the user to exploit his/her powerful visual capabilities with a view to discovering knowledge through metarules, association rules and clustering.

Visual Construction of a Target Dataset  One of the major tasks/ phases in the knowledge discovery process is the selection of a task relevant dataset. In our data mining system, the construction of a target dataset relies on two intuitive interaction spaces namely the specification space and the target space as seen in the top left part of Figure 4 as well as that of Figure 5.
Metaqueries The system provides an environment through which the user can specify patterns that the system then uses to generate corresponding metaqueries. In essence, the specification is done by linking/"joining" attributes. The user may also specify values for approximate queries and threshold levels. The user may instruct the system to search for rules that satisfy all the foregoing specifications. The system uses two main visualizations to display the results; Rules + Tuples ("Overview + Detail") and Dedicated View. Figure 4 shows the metaquerying environment.

Association Rules One of the distinct features in the environment for mining association rules is the provision of "baskets". There are two "baskets" where one corresponds to the IF part and the other represents the THEN part. The user picks an item from the target data and drops it into the relevant "basket". With regard to output, the system exploits the same visualization styles used for visualizing metarules.

Clustering The Data Mining system provides an interaction environment with various input widgets through which the user can specify parameters characterizing a clustering task. The system provides two principal visualizations for viewing clustering output; Clusters + Details ("Overview + Detail") and Separation View. Figure 5 shows the clustering environment.
Similarity Queries The system supports similarity queries over both raw data sources and clustering output data. Specification of approximate similarity queries requires a number of parameters: The query point can be easily obtained from the visualization of the target dataset or of the clustering output, by simply clicking on a point. The system provides an intuitive interface for the user to input the remaining parameters, i.e. sliders for the accuracy $\varepsilon$ and the reliability $\delta$ parameters and text boxes for the query radius (or the number of requested neighbors). Visualization of the query result relies on the ("Overview + Detail") view used for clustering (see Figure 5).

Usability As a way of getting started on usability, we carried out an informal evaluation of the interface based on standard interface design principles ("usability heuristics"). We also performed some informal user tests on a previous version of the prototype with data mining experts. We got encouraging results from the tests and even suggestions on how to improve the interface. For instance, the data mining experts suggested that the interface should provide an optional interaction environment specifically designed for the expert user and still leave the user with the freedom to switch between the two.
4 Conclusions

We have proposed an architecture for integrated data mining and presentation tool which integrates data sources, data mining algorithms and the presentation environment in a modular way. The architecture is based on the concept of metadata repository and the employment of open XML-based interchange formats to allow for easy integration of new modules for mining tasks and visualization. The implementation of a complete system based on the proposed architecture is underway. The completed system will be validated and subject to a usability study.

References


