Proceedings

PANDA Workshop
on
Pattern-Base Management Systems

April 10th, 2003
Villa Olmo, Como, Italy

Scientific Coordinators:

Yannis Theodoridis (CTI and University of Piraeus, Greece)
Michalis Vazirgiannis (Athens Univ. of Economics and Business, Greece)

Organizing Committee:

Barbara Catania (University of Milan, Italy)
Stefano Rizzi (University of Bologna, Italy)

Supported by the Commission of the European Communities under the Information Society Technologies (IST) Programme – Future and Emerging Technologies (FET), Project IST-2001-33058 PANDA

Table of Contents

PANDA Workshop on Pattern-Base Management Systems
April, 10th 2003
Como, Italy

Motivating Pattern Management

M. Vazirgiannis, E. Vrachnos and M.Halkidi

Department of Informatics
Athens University of Economics & Business
Patision 76, 10434, Athens, Greece (Hellas)
e-mail: {mvazirg, evry , mhalk}@aueb.gr

Abstract. As the amount of data produced by data processing systems increases, it has become clear that users of decision support systems do not want massive volumes of data, but are more interested in *informative* patterns buried within data. The patterns that emerge from data intensive applications have to be stored in a way so that they can be queried effectively and efficiently. Thus, we need to create pattern repositories where the knowledge artefacts known as patterns will reside, as primitive data reside in Data Base Management Systems (DBMSs). In this paper we present some representative pattern application domains that call for integrated and efficient pattern base support, along with the corresponding types of the patterns they produce. We also motivate research towards a pattern base management system by abstracting common features of patterns and management requirements

1 Introduction

The technological advances in databases have made it easy to produce and store huge volumes of data. The challenge of 21st century is the efficient and effective manipulation of these data and the extraction and management of useful knowledge from them. Sophisticated data processing tools, based on data mining, pattern recognition and other advanced knowledge extraction techniques produce huge volumes of knowledge artifacts such as association rules, clustering schemes, decision tree rules, frequent parts of a signal or text and many others. We call these artifacts *patterns.* A *pattern* represents huge volumes of information in an effective way and constitutes the cornerstone of our approach. In other words, it can be seen as a concise representation of a large set of raw data.

The patterns that emerge from data intensive applications have to be stored in a way so that they can be queried effectively and efficiently. Thus, we need to create pattern repositories where the knowledge artefacts known as patterns will reside, as primitive data reside in Data Base Management Systems (DBMSs). We call this new class of DBMS Pattern Base Management Systems (PBMS) and we address in this paper some representative application domains where PBMS may be used and produce some extraordinary results.

Another important issue when we have to manage huge volumes of data is their quality, which is critical in data analysis. In [GAN99] Ganti et al. develop a framework for quantifying the difference, between data sets in terms of the models they induce. Thus, by comparing two patterns we actually compare the (*interesting*) data characteristics of the data sets they represent. The question here is how we compare two patterns. The answer lies in the pattern representation we adopt in this paper and is called *measure component* of a pattern. The measure component constitutes a way of computing the quality of a pattern and consequently the quality of the data set it represents. In that way we can draw conclusions about the properties of the data set and "label" the data set as an important data set with respect to the information that is buried in it.

The volume of extracted patterns from various knowledge discovery applications is increasing rapidly and so does the need for effective and efficient pattern management. There are no tools or systems that are designed to deal exclusively with this problem. All current efforts for modelling patterns build an additional layer on top of a DBMS or extend a pre-existing DBMS to support patterns. This approach doesn't take advantage of the special properties of patterns and treats them just like primitive data. If we consider raw data as facts then a pattern constitutes the rules that hold for these facts, or in other words the knowledge that is hidden in the data. Thus, an ordinary DBMS is not adequate for the management of this kind of information not only because of its special structure and properties, but furthermore because of its diversity. Patterns can be clustering schemes, association rules, classification rules, probabilistic rules and many others. Moreover patterns can be found in images, signals, text, music and of course in the world wide web.

The first objective of this paper is to convince for the necessity of pattern management and the design and development of Pattern Base Management Systems. The second objective is to present certain categories of patterns in various application domains and examine their structure. Understanding the structure of patterns is essential for the design of a PBMS since a general model should be developed for the representation of all these artefacts in both logical and physical level of a PBMS.

2 Application domains generating patterns

In this section we will briefly outline application domains dealing with voluminous and heterogeneous data (further called raw data). *P*attern *ex*traction *a*lgorithms (*pexa*s) applied to these data generate potentially large quantities of patterns that essentially represent the knowledge hidden in the raw data and due to their volume and diversity call for efficient management.

a. Data Mining

Data Mining is mainly concerned with methodologies for extracting patterns from large data repositories. The extracted patterns are evaluated based on some interestingness measures that identify patterns representing *interesting* knowledge, i.e., *interesting patterns*. Then, we could adopt a broader view of data mining functionality, considering data mining as the process of discovering interesting knowledge from large amounts of data stored in databases, data warehouses or other information repositories. There are many different data mining algorithms that produce a variety of patterns for a certain data set. We give a brief description of the patterns that are produced by these algorithms.

Clustering

Clustering is the process of partitioning a given data set into groups (clusters) such that the data points in a cluster are more similar to each other than points in different clusters [GRS98]. The clustering process may result in different partitions of a data set, depending on the criterions used for clustering. Then we may evaluate the quality of each clustering scheme and make the selection that meets our needs. Clustering is valuable in many fields.

Classification – Decision Making

The classification problem has been studied extensively in statistics, pattern recognition and machine learning community as a possible solution to the knowledge acquisition or knowledge extraction problem [RS98]. A number of classification techniques have been developed and are available in bibliography. Among these, the most popular are: *Bayesian classification*, *Neural Networks* and *Decision Trees*. As we have already discussed classification is a form of data analysis that can be used to define data models describing data classes or predict data trends. It can be used for making intelligent bases decisions in business and science.

Association Rules

Association rules reveal underlying interactions between attributes in the data set. These interactions can be presented with the following form: $A \rightarrow B$, where A, B refer to sets of attributes' values in underlying data. More specifically, A and B are selected so as to be frequent item sets. A formal statement of the problem can be found in [AS94]. The intuitive meaning of such a rule is that records in the dataset, which contain the attributes in A, tend also to contain the attributes in B [SA95]. We note also that the extracted rules have to satisfy some user-defined thresholds related with association rules measures (such as support, confidence, leverage, lift). A typical application of association rule mining is market basket analysis. This process analyses customer-buying habits by finding associations between the different items that customers place in their "shopping baskets". The discovery of such associations can help retailers develop marketing strategies by finding which items are frequently purchased together by customers.

b. Sequential Patterns- Time Series Analysis

Sequential pattern mining is the mining of frequently occurring patterns related to time or other sequences. Most studies on sequential pattern mining concentrate on symbolic patterns. The problem of mining sequential patterns can be stated as follows:

Given a potentially large pattern (string) S, we are interested in sequential patterns of the form $a \rightarrow b$, where a, b are substrings inside S, such that the frequency of ab is not less than some minimum support and the probability that a is immediately followed by b is not less than minimum confidence.In daily and scientific life sequential data are available and used everywhere. Some representative examples are text, music notes, weather data, satellite data streams, business transactions, telecommunications records, experimental runs, DNA sequences, histories of medical records. In [WAN97] we find a motivation for sequential pattern databases: *Since the underlying database is usually large, dealing with changing data and patterns is a challenge for research and application in knowledge discovery and data mining, and incremental methods for updating the patterns are possible solutions*.

c. Signal Processing: Content-Based Music Retrieval [V+02]

Large-scale storage of sound and music has only become possible in the last decade. In addition, the new possibility for wide-area distribution of multimedia over the Internet has given rise to new requirements for flexible and powerful databases for musical and audio data. One of these requirements is the complexity of a selection query upon a database that contains massive amounts of musical data [PAC00]. For example consider the following question: "I want to browse through Bach Fugues recorded in C minor and performed with a clavichord". It is clear that this kind of queries address information hidden in the content of the music signal and raise the following challenges related to content-based music retrieval: instrument recognition, melody spotting, musical key extraction, musical pattern recognition, composer recognition, music structure extraction and music segmentation, to name but a few.

d. Patterns in Information Retrieval [V+02]

Another research field where patterns are apparent is that of Information Retrieval. In a retrieval setting we have a collection of discourse material, also called *corpus*, and users submit queries to the system in order to retrieve information that suits their interest. Queries are often vaguely defined, in contrast to traditional database systems, due to lack of a query language or algebra. A query consisting of only a few words does not always reflect the user's actual interest; therefore users often experience frustration from a retrieval system. The *Latent Semantic Indexing* (LSI) [DDF+90, BDO95], a retrieval model, unveiling patterns in terms' usage, seems to produce more effective information retrieval.

e. Mathematics

Mathematics is the science of patterns. Patterns arise in the form self correlation mappings in a data set (i.e. consider the Fibonacci equation: $y_{n+1} = y_n \; y_{n-1}$, $y_0 =0$, $y_0=1$) or in the form the analytical polynomial expressions bounding independent variables others (i.e. $y=ax^2+bx+c$). As we advance, we experience number patterns again through the huge concept of functions in mathematics. But patterns are much broader. They can be sequential [AS95], spatial [ST99], temporal [DLM+98][SB98], and even linguistic [FFK+98][LAS97]. Sometimes it is very useful to collect number patterns so as to be able to learn the behaviour of numbers collected by telecommunication data for example [BCH00] or by equation solving [SB99].

Shape patterns

The concept in this case is that the majority of geometrical shapes share a structural similarity. For example the number of vertices constituting a polygon etc. The existence of such patterns can help us in recognizing familiar shapes in image processing [NPP00] or even for reconstructing polygonal images [CN95].

Patterns in Cryptography

Cryptography is one of the main subjects in which mathematics rays supreme. In cryptography, every cryptographic system could be considered as a pattern itself. For example, if we have many sets of raw data and in each set we enforce a specific encryption, then these sets would share the similarity of their encryption attribute. We can find many such cases in cryptography, like *Vigenere Cipher*, *Caesar Cipher*, *Gronsfeld cipher* etc. Such Cryptographic patterns are widely used in word processors, electronic commerce systems, spreadsheets, databases and security systems [BRD99].

3 Need for pattern management – a wish list…

In this section we will attempt an abstraction of facts and notions common among the previously mentioned application domains. The pattern extraction phase results in patterns extracted from a raw data set using one or more pexas. It is clear that from a raw data set A multiple a. types and b. instances of a pattern type can be extracted. Different types, as a result of different pexa *type* exploitation (i.e. both clusters and association rules can be extracted from A). Different instances because of i. the usage of different pexas of the same type (i.e. different clustering algorithms on A result in different partitioning schemes of the same dataset) or ii. because of the application of the same pexa with different input parameters (again, the example arising is for a clustering algorithm multiple sessions on the same data set each time with different input parameters potentially resulting in different partitions).

In all cases the pexa forms a *mapping* between the *raw data* space and the *pattern* space. In most of the cases the pattern is either:

- a *repeating subset of the raw data set* either in a. terms of *structure* (i.e. a record structure {name, address, salary}) or in terms of *values* (i.e. the association rule buys(beer, diapers)->buys(snacks), or the constraint: <salary <=4000 Euro >.

- or *a repeating relationship between parts of the raw dataset*. Such are a. the analytical form of a polynomial constraint holding true in a subset of the raw data set, b. grouping together multiple raw dataset points under a cluster label as result of a clustering algorithm.

Patterns are grouped into pattern types. A pattern type usually involves a i. *structure* (representing its syntactic form), ii. a *source description*, representing the raw dataset specification iii. *a set of measures*, representing the instantiation of its structure with regards to a raw data set (conforming to the source description) and iv. an algorithm type, representing the pexa type and bearing the semantics of the pattern.

It is clear that from a raw dataset multiple pattern types can be extracted due to the application of diverse pexas. For instance from a music score both pitch and rhythm patterns can be extracted. Assuming thus a wealth of pattern types and instances (forming the pattern base of the specific raw dataset) that are extracted from the raw dataset A and given the important resources consumed for the extraction of these patterns one would argue that it would be necessary to:

- store and reuse these patterns in order to fulfill requirements of the users for later decision making
- define a valid mapping between the pattern base and the raw data in order to be able to switch between raw data and their pattern base.

In order to achieve these objectives it is necessary to define a pattern base management system that will efficiently represent, store and retrieve patterns and communicate with the raw data repository. One would argue that mature relational/object-relational or object oriented approaches could facilitate these requirements. We will confront this argument:

- Patterns are fundamentally different to the usual alphanumeric databases as their structure can be much more complex as well as diverse (i.e. compare the structure and semantics if time series patterns to clusters or association rules).
- Pattern semantics are much richer than the raw dataset ones and are usually implied by the knowledge of the pexa.
- Patterns' behavior/functionality is significantly more complex (i.e. treating a polynomial constraint). This involves also the similarity issue that is becoming much more complex and involves multiples dimensions of similarity, such as i. intra-pattern vs. inter-pattern similarity ii. structural vs. value based similarity etc.

As it is clear from the previous, the diversity of patterns in terms of structure, values and behavior call for a fundamentally different approach on representation storage and management both in logical and physical level. This implies the definition of appropriate *modeling structures* that will be mapped to appropriate *physical* ones. Also there is a clear need for a proper set of *operators* that represent the semantics of the pattern behavior and manipulation (the limited SQL manipulation actions are apparently inadequate here). These operators will populate the envisaged *Pattern Query Language* (PQL) that will deal both with definition and manipulation. Consequently, in order to make pattern based queries efficient, the appropriate *pattern indexing structures* should be designed.

4 Conclusion

In this paper we briefly reviewed some of the application domains that need pattern base support and motivate the need for a PBMS. In all these fields various pexas are applied on potentially huge and diverse data repositories. The knowledge artefacts that being produced, constitute a concise representation of the repository known as its **pattern base**. These artefacts that we call patterns need to be treated as persistent objects that can be stored, accessed and queried effectively and efficiently. All the approaches that have been proposed and implemented so far, build an additional level on top of a DBMS, and don't take advantage of the special properties of patterns such as the fact that they constitute compact and rich in semantics representations of raw data.

The efficient and effective manipulation of pattern bases imposes a new class of DBMSs, namely **Pattern Base Management Systems** that treat patterns just like DBMS treat raw data. Thus, patterns are modelled both in logical and physical level, they are stored, visualized, indexed and queried.

References

[WAN97] Wang K. "*Discovering Patterns from Large and Dynamic Sequential Data*", in Journal of Intelligent Information Systems 9, 33-56 (1997).

[AS94] Rakesh Agrawal, Ramakrishnan Srikant. "Fast Algorithms for Mining Association Rules". Proc. of the 20th VLDB Conference, 1994.

[AS95] Rakesh Agrawal and Ramakrishnan Srikant: *Mining Sequential Patterns,* IBM Almaden Research Center, 1995.

[BRD99] Alexandre M. Braga, Cecilia M.F. Rubira and Ricardo Dahab: *Tropyc: A Pattern Language for Cryptographic Software,* Brazil, 1999.

[CN95] Peter Clifford and Geoff Nichols: A Metropolis Sampler for Polygonal Image Reconstruction, UK, 1995.

[DDF+90] S. Deerwester, S. T. Dumais, G. Furnas, Th. K. Landauer, R. Harshman, Indexing by Latent Semantic Analysis, *Journal of the Society for Information Science*, 41(6): 391-407, 1990.

[DLM+98] Gautan Das, King-Le Lin, Heikki Mannila, Gopal Renganathan and Padhraic Smith: *Rule discovery from time series,* USA, 1998.

[FFK+98] Roner Feldman, Moshe Fresko, Yakkov Kihar, Yehuda Lindell, Orly Liphstat, Mrtin Rajman, Yonatan Schler, Oren Zamiv: *Text Mining at the Term Level,* 1998.

[FPSU96] U. Fayyad, G. Piatesky-Shapiro, P. Smuth & R. Uthurusamy(editors). "From DataMining to Knowledge Discovery: An Overview". *Advances in Knowledge Discovery and Data Mining*. AAAI Press, 1996.

[GAN99] Ganti V, Ramakrishnan R., Gehrke J., and Loh W.Y. "A Framework for Measuring Differences in Data Characteristics". PODS, 1999.

[GRS98] Sudipto Guha, Rajeev Rastogi, Kyueseok Shim. "CURE: An Efficient Clustering Algorithm for Large Databases", *Published in the Proceedings of the ACM SIGMOD Conference*, 1998.

[NPP00] Chicahito Nakajima, Massimiliano Pontil and Tomaso Pogio: *People Recognition and Pose Estimation in Image Sequences*, JAPAN, 2000.

[PAC00] F. Pachet, P. Roy, D. Cazaly, "A Combinatorial approach to content-based music selection", IEEE Multimedia, Vol 1, 2000.

[REI01] J. Reiss, J. Aucouturier, M. Sandler, "Efficient Multi-dimensional searching routines for music information retrieval", Proceedings of ISMIR 2001.

[RS98] R. Rastori, K. Shim. "PUBLIC: A Decision Tree Classifier that Integrates Building and Pruning". *Proceeding of the 24th VLDB Conference,* New York, USA, 1998.

[SA95] Ramakrishnan Srikant, Rakesh Agrawal. "Mining Generalized Association Rules". Proc. of the 21st VLDB Conference, 1995.

[SB98] Neil Sumpter and Andrew J. Bulpitt: *Learning Spatio-Temporal Patterns for predicting Object behaviour,* UK, 1998.

[SB99] Stephan Schulz and Felix Brandt: *Using Term Space Maps to Capture Search Control Knowledge in Equational Theorem Proving,* Germany, 1999.

[ST99] Ayman A. Abel-Samad and Ahmed H. Tewfik*: Search Strategies for radar Target localization*, University of Minnesota, Minneapolis, 1999.

[V+02] M. Vazirgiannis et. Al., "Pattern Apllication Domains, A survey", PANDA Technical report available at: http://dke.cti.gr/panda

Current Issues in Modeling Data Mining Processes and Results

Irene Ntoutsi[1,2] and Yannis Theodoridis[1,2]

[1] Computer Technology Institute, Greece

[2] University of Piraeus, Greece

E-mail: {ntoutsi, ytheod}@cti.gr

Abstract.

So far, research and industrial community has recognized the need for standards in order to create, store and manage data mining results. These standards should be independent of the underlying data mining systems and should allow the communication between different vector applications. In this paper we address the current and evolving efforts towards this aim. As we will see there are a lot of standards and tools, with PMML being the most prominent one, however most of them deal with patterns in a traditional manner (usually with relational tables). As the amount of data produced by data processing systems increases, it has become clear the need for a system, similar to DBMS, for efficient pattern management.

1. Introduction

In this paper, we address the current and evolving efforts on modeling data mining processes and their results.

The outline of the paper is as follows: In section 2 we introduce the Predictive Model Markup Language (PMML) developed by the Data Mining Group (DMG). In section 3 we give an overview of the SQL/MM Part 6, a standard that has been developed under ISO. In section 4 we give an overview of the Common Warehouse Model and the Data Mining MetaModel that are standards supported by OMG. In section 5 we introduce the efforts of individual vendors such as Oracle, IBM, SPSS, etc. in developing the Java Data Mining API. In section 6 we provide an overview of the Oracle 9i Data Mining componentts. In section 7 with present the Pattern Query Language (PQL) used by the Information Discovery DataMining Suite. In section 8 we provide an overview of the Microsoft work on data mining. We conclude in section 9 giving a comparative presentation of the overviewed data mining standards and tools.

2. Data Mining Group / Predictive Model Markup Language

PMML - Predictive Model Markup Language [DMG] is an XML-based language, developed by DMG – Data Mining Group, which provides a quick and easy way for companies to define predictive models and share models between compliant vendors' applications. PMML provides applications a vendor-independent method of defining models so that proprietary issues and incompatibilities are no longer a barrier to the exchange of models between applications. It allows users to develop models within one vendor's application, and use other vendors' applications to visualize, analyze, evaluate or otherwise use the models.

2.1. PMML Data Flow

PMML defines a variety of specific mining models such as for tree classification, neural networks, regression, etc. Equally important there are also definitions which are common to all models, in order to describe the input data itself, and generic transformations which can be applied to the input data before the model itself is evaluated. In the following schema the basic blocks of a mining model as well as the data flow of such operation are shown:

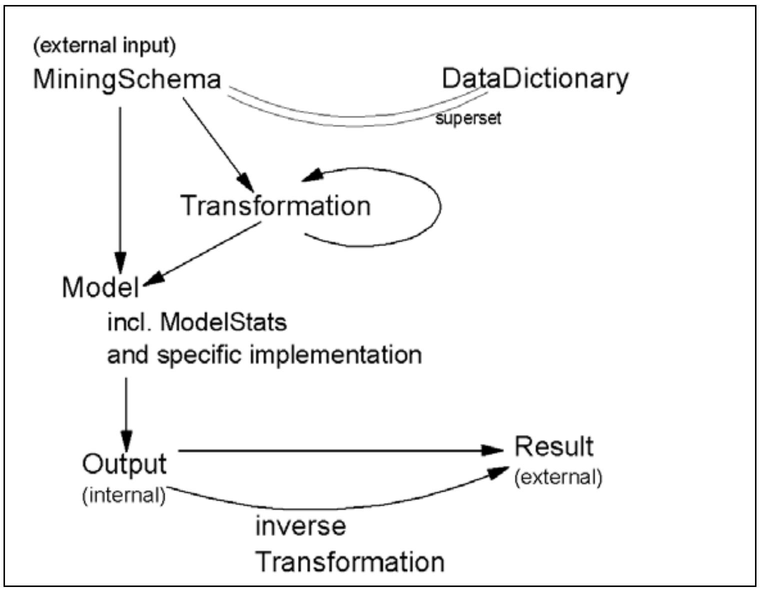


Figure 1. The data flow in PMML

- **Data Dictionary**: The DataDictionary describes the data 'as is', that is the raw input data. The DataDictionary refers to the original data and defines how the mining model interprets the data, e.g., as categorical, or numerical, and the range of valid values may be restricted. The raw data are not included in a PMML document and they are hosted in external sources. The DataDictionary only defines the mappings between the source attributes and the model's local field names.
- **MiningSchema**: The MiningSchema defines an interface to the user of PMML models. It lists all fields that are used as input to the computations in the mining model. The mining model may internally require further derived values that depend on the input values, but these derived values are not part of the MiningSchema. The derived values are defined in the transformations block. The MiningSchema also defines which values are regarded as outliers, which weighting is applied to a field, e.g., for clustering. Input fields as specified in the MiningSchema refer to fields in the data dictionary but not to derived fields because a user of a model is not required to perform the normalizations.
- **Transformations**: Various types of transformations are defined such as normalization of numbers to a range [0..1] or discretization of continuous fields. These transformations convert the original values to internal values as they are required by the mining model such as an input neuron of a network model. If a PMML model contains transformations a user is not required to take care of these normalizations. The MiningSchema lists the input fields that refer to the non-normalized original values, the user presents these fields as input to the model. If a PMML model contains transformations a user is not required to take care of these normalizations. The MiningSchema list the input fields which refer to the non-normalized original values, the user presents these fields as input to the model. The transformations in PMML are intended to cover expressions that were generated by a mining technique. A complete mining project usually needs many other preprocessing steps which may have to be defined manually. PMML 2.0 does not provide a complete language for this full preprocessing. These data preparations steps must be performed before feeding the values into a PMML consumer.
- **Model**: PMML defines a variety of specific mining models such as for tree classification, neural networks, regression, etc. The specific definitions of may refer to fields listed in the MiningSchema or to derived fields which can be computed from the MiningSchema-fields.
- **Output**: The output of a model always depends on the specific kind of model, e.g. it may by defined by a leaf node in a tree or by output neurons in a neural network.
- **Result**: The final result, such as a predicted class and a probability, are computed from the output of the model. If a neural network is used for predicting numeric values then the output value of the

network usually needs to be denormalized into the original domain of values. Fortunately, this denormalization can use the same kind of transformation types. The PMML consumer system will automatically compute the inverse mapping.

Since PMML is an XML basedstandard, the specification comes in the form of an XML Document Type Definition (DTD). In the next sections we will present the main features of the PMML.

2.2. General Structure of a PMML Document

As we already mentioned PMML uses XML to represent mining models. The structure of the models is described by a DTD which is called the PMML DTD. One or more mining models can be contained in a PMML document. A PMML document is an XML document with a root element of type PMML.The DTD that all PMML documents must conform is:

```
<?xml version="1.0"?>

   <!DOCTYPE PMML PUBLIC "PMML 2.0" "http://www.dmg.org/PMML2.0/pmml-2-0.dtd">

   <PMML  version="2.0">

         ...

   </PMML>
```

For PMML version 2.0 the attribute version must have the value 2.0.

A PMML document is not required to have a declaration. If there is one then a PMML document must not depend on external parameters. Although a PMML document must be valid with respect to the PMML DTD, a document must not require a validating parser, which would load external entities. In addition to being a valid XML document, a valid PMML document must obey a number of further rules which are described in the PMML specification. The root element of a PMML document must have type PMML.

```
<!ENTITY % A-PMML-MODEL '(TreeModel | NeuralNetwork | ClusteringModel |

                               RegressionModel | GeneralRegressionModel |

                               NaiveBayesModels | AssociationModel |

                               SequenceMiningModel )' >

   <!ELEMENT PMML ( Header, Settings?, DataDictionary,

                      TransformationDictionary, (%A-PMML-MODEL;)+, Extension* )>


   <!ATTLIST PMML version CDATA #REQUIRED>

   <!ELEMENT Settings (Extension*) >

   <!ELEMENT TransformationDictionary (DerivedValues*, Extension* ) >
```

The basic elements of an PMML document are: Header, Settings, Data Dictionary and Transformation Dictionary (Derived Values).

3. SQL/MM

SQL/MM (MM for MultiMedia) [SQL/MM] is a standard based on SQL that has been developed by the International Organization for Standardization (ISO). It is divided into the following parts: Part 1 (Framework), Part 2 (Full Text), Part 3 (Spatial), Part 5 (Still Image), Part 6 (Data Mining).

The structured types defined in SQL/MM are *first-class SQL types* that can be accessed through SQL:1999 base syntax. These accesses also include invocation of the routines (methods) associated with the structured types. In the following, we focus on the basic features of Part 6, Data Mining.

3.1. SQL/MM Data Mining

The standard supports four different data mining techniques. The term *model* is used throughout the standard and it actually stands for *data mining technique*. The four models are: Rule model, Clustering model, Regression model and Classification model.

Every model has a corresponding SQL structured user-defined type. A set of predefined types completes the full definition of each model. The basic type is named *DM_*Model* where "*" is replaced by 'Class' for a classification model, 'Rule' for a rule model, 'Clustering' for a clustering model and 'Regression' for a regression model.

The predefined types are the following (the same naming schema is used as in the case of the DM_*Model):

– DM_*Settings: Instances of that type are used for storing various parameters of the data mining model, such as the maximum number of clusters or the depth of a decision tree.

– DM_*TestResult: Instantiations of that type hold the results of the testings during the training phase of the data mining models.

– DM_*Result: The running of a data mining model against real data creates instances of that type.

– DM_*Task: Instances of that type store metadata that describe the process and control of testings and of the actual runnings.

4. Common Warehouse MetaModel

The CWM™ - Common Warehouse Metamodel [CWM] is a specification that describes metadata interchange among data warehousing, business intelligence, knowledge management and portal technologies. The main purpose of CWM is to enable easy interchange of warehouse and business intelligence metadata between warehouse tools, warehouse platforms and warehouse metadata repositories in distributed heterogeneous environments.

CWM is based on three key industry standards:

– *UML - Unified Modeling La*nguage, an OMG modeling standard.

– *MOF - Meta Object Facility*, an OMG metamodeling and metadata repository standard that bridges the gap between dissimilar meta-models by providing a common basis for meta-models. If two different meta-models are both MOF-conformant, then models based on them can reside in the same repository.

– *XMI - XML Metadata Interchange*, an OMG metadata interchange standard.

The CWM provides a framework for representing metadata about data sources, data targets, transformations and analysis, and the processes and operations that create and manage warehouse data and provide lineage information about its use.The CWM Metamodel constists of a number of sub-metamodels which represent common warehouse metadata in major areas of interest to data warehousing and business intelligence, icluding Data Resources, Data Analysis and Warehouse Management.

The CWM is designed to maximize the reuse of Object Model (a subset of UML) and the sharing of common modeling constructs where possible. The most prominent example is that CWM reuses/depends on Object Model for representing object-oriented data resources.

Bellow we give an overview of the Data Mining sub-metamodel.

4.1. Data Mining Metamodel

The CWM Data Mining sub-metamodel represents three conceptual areas: the overall Model description itself, Settings and Attributes.

The Model conceptual area consists of a generic representation of a data mining model (that is, a mathematical model produced or generated by the execution of a data mining algorithm). This consists of:

– *MiningModel*, a representation of the mining model itself

– *MiningSettings*, which drive the construction of the model

– *ApplicationInputSpecification*, which specifies the set of input attributes for the model

– *MiningModelResult*, which represents the result set produced by the testing or application of a generated model.

The **Settings** conceptual area elaborates further on the Mining Settings and their usage relationships to the attributes of the input specification. Mining Settings has four subclasses representing settings for:

– *StatisticsSettings*

– *ClusteringSettings*

– *SupervisedMiningSettings*

– *AssociationRulesSettings.*

The SupervisedMiningSettings are further subclassed as ClassificationSettings and RegressionSettings, and a CostMatrix is defined for representing cost values associated with misclassifications.

The **Attributes** conceptual area defines two subclasses of Mining Attribute:

– *NumericAttribute*

– *CategoricalAttribute*. A categorical attribute might possess some Category property and might be associated with some kind of taxonomy with a CategoryHierarchy.

5. Java DM API

Java DM API (JDMAPI) [JDMAPI] follows SUN's Java Community Process as a Java Specification Request (JSR). It addresses the need for an independ of the underlying data mining system API that will support the creation, storage, access and maintenance of data and metadata supporting data mining models, data scoring, data mining results, and data transformations.

Refering to the underlying technologies the JDMAPI will be based on a highly-generalized, object-oriented, data mining conceptual model leveraging emerging data mining standards such Common Warehouse Metamodel (CWM), ISO's SQL/MM for Data Mining, and DMG's PMML.

The JDMAPI model will support four conceptual areas that are generally of key interest to users of data mining systems: settings, models, transformations, and results. The JDMAPI specification does not prescribe any particular implementation strategy. Vendors will decide if they will implement JDMAPI as the native API of their product or if they will develop a driver/adapter that mediates between a core JDMAPI layer and multiple vendor products.

JDMAPI has three logical components that may be implemented as one executable or in a distributed environment:

– *Application Programming Interface* (API) - The API is the end-user-visible compo-nent of a JDMAPI implementation that allows access to services provided by the data mining engine (DME).

– *Data Mining Engine* (DME) – The DME provides the infrastructure that offers a set of data mining services to its API clients. When implemented as a server of a client-server architecture, it is referred to as a data mining server (DMS).

– *Metadata Repository* (MR) - The DME uses a metadata repository which serves to persist data mining objects. This repository can be based on the CWM framework, specifically leveraging the

CWM Data Mining metamodel, or implemented using a vendor proprietary representation. The MR may exist in a file-based environment, or in a relational database.

The ultimate goal of JDMAPI is to provide for data mining systems what JDBC did for relational databases. The version 0.91 of the JDMAPI has been released for public review since November 2002.

6. Oracle9i Data Mining

Oracle9i Data Mining [Oracle9i] provides comprehensive data mining functionality that is embedded in the Oracle9i Database.

Oracle9i Data Mining has two main components:

- *Oracle9i Data Mining API*. The ODM API provides an early look at concepts and approaches being proposed for JDMAPI (Section 5). Ultimately, Oracle9i Data Mining will comply with the JDMAPI standard after it is published.
- *Data Mining Server (DMS)*. The Data Mining Server (DMS) is the server-side, in-database component that accepts requests from programs written using the ODM API, process these requests and delivers results to the client applications. It also provides a metadata repository consisting of mining input objects and result objects, along with the namespaces within which these objects are stored and retrieved.

Oracle9i Data Mining supports the following data mining functions: Classification (supervised learning), Clustering (unsupervised learning), Association Rules (unsupervised learning), Attribute Importance / Feature selection (supervised learning)

Oracle9i Data Mining supports the following data mining algorithms: Adaptive Bayes Network supporting decision trees (classification), Naive Bayes (classification), Model Seeker (classification), k-Means (clustering), O-Cluster (clustering), Predictive variance (attribute importance), Apriori (association rules)

Every model has characteristics similar to those addressed in PMML and SQL/MM Data Mining. Models are build and stored in the DMS and the users can refer to them using a user-specified unique name. Every step in the model building process creates persistent objects that can be used from multiple applications. After a model is build, model testing estimates the accuracy of a model's predictions and test results are stored in a mining test result object.

The most important object, in context of storing data mining results, is the mining result object. The mining result object contains the end products of one of the following mining tasks: build, test, compute lift, or apply. ODM supports the persistence of mining results as independent, named entities (relational tables) in the DMS. A mining result object contains the operation start time and end time, the name of the model used, input data location, and output data location for the data mining operation.

Oracle intends to provide full support of SQL/MM and PMML in future versions.

7. Information Discovery DataMining Suite

Information Discovery has developed a set of tools and systems for data mining and knowledge extraction [IDMS]. The output of the Data Mining Suite is stored within a pattern-base and is accessible trough PQL: The Pattern Query Language^{TM}, a new language specifically designed by the company for business user decision support. PQL is a pattern-oriented query language specifically designed to provide business users access to refined information, just as SQL provides access to data. PQL resembles SQL, but is implemented above SQL with additional C constructs. The PQL Pattern Kernel consistently merges and manages multiple pattern-types.

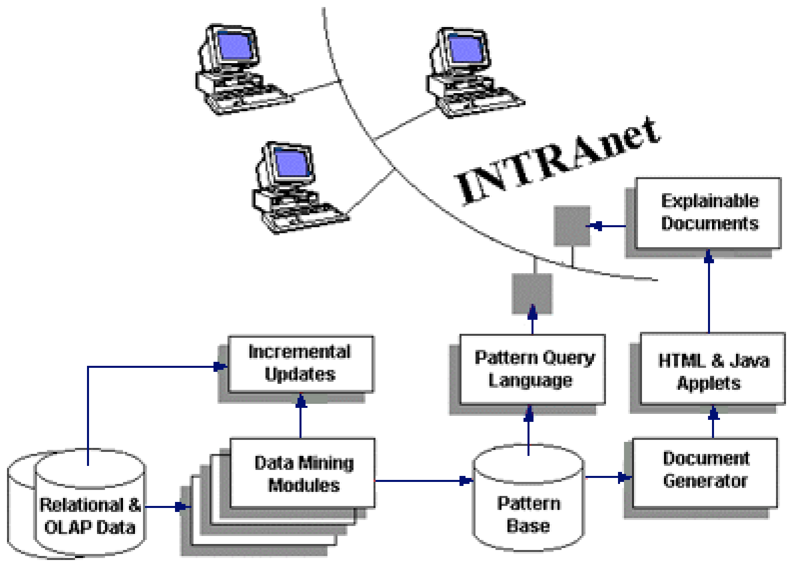


Figure 2. The IDMS approach (figure taken from [IDMS])

Patterns discovered by the Data Mining Suite are rule-based. Discovery takes place simultaneously along multiple dimensions on the server and the results are delivered to the user in plain English, accompanied by tables and graphs that highlight the key patterns.

No other information is available (at least through their web site) about the syntax or the semantics of the language, neither about the way patterns are stored in the pattern-base. However, its notion is very close to the motive of the PANDA project [PANDA] and that is the reason for mentioning it in this paper.

8. Microsoft Data Mining

Microsoft introduced Data Mining in SQL Server™ 2000 Analysis Services and initiated the OLE DB API for Data Mining [Microsoft1]. OLE DB for Data Mining serves as a standard that external product vendors can use for delivering their data-mining functionality in the Microsoft environment. This API is supported by a number of leading data mining providers including Megaputer Intelligence Inc, SPSS Inc etc.

The OLE DB API for Data Mining encapsulates the idea of a universal data-access mechanism that allows the sharing of data and data-mining results through heterogeneous environments with multiple applications.

Some of the basic OLE DB API features include:

– *Data-mining model*. The data-mining model is like a relational table, except that its columns can be used to derive the patterns and relationships that characterize the data mining results. These columns can also be used for predictions; the data-mining model serves as the core functionality that both creates a prediction model and generates predictions. Unlike a standard relational table, which stores raw data, the data-mining model stores the patterns discovered by the data-mining algorithm.

A data mining model is created throughn a CREATE statement, similarly to the SQL CREATE TABLE statement. A data data mining model is populated by using the INSERT INTO statement, just as in the case of a table population. The client application issues a SELECT statement to make predictions through the data-mining model. After the mining engine defines the important fields (traing phase) and stores them in the data-mining model, the model can use the same pattern to classify new data in which the outcome is unknown. The trained pattern, or structure, is saved in the data-mining model.

```
CREATE MINING MODEL [Age Prediction]
(
         [Customer ID]                LONG      KEY,
         [Gender]                     TEXT      DISCRETE,
         [Age]                        DOUBLE    DISCRETIZED() PREDICT,
         [Product Purchases]          TABLE
         (
                  [Product Name]      TEXT      KEY,
                  [Quantity]          DOUBLE    NORMAL CONTINUOUS,
                  [Product Type]      TEXT      DISCRETE RELATED TO [Product Name]
         )
)
USING [Decision Trees]
```

– *Prediction join operation*. The OLE DB API defines a data mining query language based on SQL syntax. This operation is mapped to a join query between a data-mining model (which contains the trained pattern from the original data) and the designated new input data.

– *Predictive Model Markup Language* (PMML). The OLE DB for Data Mining specification incorporates the PMML standards of the Data Mining Group (see Section 2).

Microsoft SQL Server 2000 Analysis Services supports two data mining tasks: classification (with decision trees) and clustering (with k-Means) [Microsoft2]. However, the coming version of SQL Server is expected to include more data mining algorithms.

9. A comparison of data mining standards and tools and conclusions

In Tables 1 and 2, we compare the data mining standards and tools, respectively, overviewed so far.

Comparatively, PMML looks to be a step ahead from the rest standards and, since it is XML-based, it could be more easily adopted by big 'players'. On the other hand, storage in relational tables is an inherent shortcoming (like for any XML database stored in relational DBMS).

The questionmarks in both tables indicate that vendors' work is in progress with several issues being still open as well as that details of their work are not public at this time.

To our opinion, two issues make clear that the road is still far from the efficient pattern management:

– *Storage structure for mining results*: all proposals converge in using relational tables for storing mining results. This could be a straightforward solution for vendors of commercial relational or object-relational DBMS but definitely needs to be revisited since mining results are far from traditional.

– *Extensibility for novel mining models*: a user is restricted to choose among supported mining models (a limited number in any of the above tools) and no support is given when he/she asks for a novel model. This might be very restrictive in several applications.

Concluding, the need for efficient pattern management is obvious and drives the data mining community research and applications domain. So far, there have been a lot of efforts towards this aim, with PMML being the most popular approach. However all these efforts follow the traditional object-relational approach and do not offer a complete solution to the problem of efficient pattern manipulation.

Table 1. Comparison of data mining standards

	PMML	**SQL/MM**	**CWM**	**JDMAPI**	**Microsoft OLE DB API**
Based on	XML	SQL	UML MOF XMI - XML	CWM SQL/MM PMML	SQL PMML
Programming language	XML like	SQL like	UML like	Java like	SQL like
Querying mining results	XML like	SQL like	?	?	SQL like

Table 2. Comparison of data mining tools

	Oracle9i Data Mining	**Information Discovery DataMining Suite**	**Microsoft SQL Server Analysis Services**
Supported standards	JDMAPI PMML SQL/MM	?	OLE DB API for Data Mining PMML
Storage structure for mining results	Relational tables	Relational tables	Relational tables
Querying mining results	?	PQL – Pattern Query Language (resembles SQL)	SQL-based data mining query language
Supported mining operations (algorithms)	Classification (Adaptive / Naïve Bayes, Model Seeker) Clustering (k-Means, O-Cluster) Association Rules (Apriori)	?	Classification (Decision Trees) Clustering (k-Means)
Building mining models	The user defines the input data and the mining function settings (a user is allowed to specify the desired types of results from existing algorithms)	?	The user defines the structure (keys, columns) and properties (specific algorithm and parameters) of the model
Import/export facilities	Imports/ exports PMML for Naïve Bayes Classification and Association Rules	?	?

References

[CWM] Common Warehouse Metamodel (CWM), available at http://www.omg.org/cwm/ (valid as on April 2, 2003).

[DMG] DMG, Predictive Model Markup Language (PMML), available at http://www.dmg.org/pmmlspecs_v2/pmml_v2_0.html (valid as on April 2, 2003).

[IDMS] Information Discovery DataMining Suite, available at http://www.patternwarehouse.com/dmsuite.htm (valid as on April 2, 2003).

[JDMAPI] Java Data Mining API, available at http://www.jcp.org/aboutJava/communityprocess/review/jsr073/ (valid as on April 2, 2003).

[Microsoft1] Data Mining in SQL Server 2000, available at http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnsqlmag01/html/datamining.asp (valid as on April 2, 2003).

[Microsoft2] Microsoft SQL Server - Performance Study of Microsoft Data Mining Algorithms, available at http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnsql2k/html/ sql2k_analysisdm.asp (valid as on April 2, 2003).

[Oracle9i] Oracle9i Data Mining Concepts, available at http://otn.oracle.com/docs/products/oracle9i/doc_library/release2/datamine.920/a95961/1concept.htm (valid as on April 2, 2003).

[PANDA] Patterns for Next-generation Database Systems, IST/FET Project. Project's web site available at http://dke.cti.gr/panda (valid as on April 2, 2003).

[SQL/MM] ISO SQL/MM Part 6, available at http://www.wiscorp.com/SQLStandards.html#sqlmmreadings (valid as on April 2, 2003).

Acknowledgment:
This research is partially supported by the Greek Ministry of Education and the European Union under a grant of the "Heracletos" EPEAEK II Programme (2003-06)

Architecture for Pattern-Base Management Systems

Manolis Terrovitis

National Technical University of Athens,
Dept. of Electrical and Computer Eng.,
Athens, Hellas
mter@dbnet.ece.ntua.gr

Panos Vassiliadis

University of Ioannina,
Dept. of Computer Science,
Ioannina, Hellas
pvassil@cs.uoi.gr

Abstract. In many modern applications we have to deal with huge volumes of data. Many techniques have been developed on how to extract knowledge, statistical usually, from them, especially in the context of data mining. The results of such operations are abstract and compact representations of the original data, which we name *patterns*. Still, these patterns have to be further elaborated to be used in an effective way. In this paper we present the architecture of a *Pattern Base Management System* that can be used to efficiently store, and query patterns. We present its logical structure and we comment on the criteria of whether the existing systems for storing and manipulating data can cover the special user requirements that patterns impose.

1. Introduction

Today's world produces an enormous amount of data in a regular basis. Huge amounts of *raw data* have to be processed and taken into consideration before new solutions and decisions are made in several fields. Raw data are facts that have an implicit meaning and have been recorded from various sources in the real world. This recording can be done by humans through data entry procedures, or by collecting measurements from various instruments or devices, e.g., cellular phones, environment measurements, monitoring of computer systems, etc. (Figure 1). The determining property of raw data is the vastness of their volume; moreover, a certain degree of heterogeneity is also present.

Clearly, data in such huge volumes do not constitute *knowledge* per se; i.e., they cannot be directly exploited by human beings and no useful information can be deduced simply by their observation. Thus, more elaborate techniques are required in order to extract the hidden knowledge and make these data valuable to the end-users. Data mining [BeLi96, FPSU96, HaKa01, Vaz+02] was developed to help extract knowledge from the raw data, using algorithms that could discover several statistic properties in the original data. Data mining produces results like association rules, clusters, decision trees and other structures that describe properties of the raw data. The common characteristic of all these techniques is that big portions of the available data are abstracted and represented by a small number of knowledge-carrying representatives, which we call *patterns*. Patterns in this sense do not appear only in data mining but in several other fields, like signal processing, image recognition, spatial databases and others.

The patterns despite being already the results of some elaboration on the raw data, are not, usually, in a form that can lead us directly to real life results. We need tools that will permit us to compare, query and store the patterns in order to retrieve the information we want. This paper describes the architecture of a system named *Pattern Database Management System* (PBMS) that will provide such tools for pattern manipulation.

The main focus of this work is dual: (a) it describes motivation behind and the architecture of the PBMS, and (b) it explains why such a system in needed and why no existing system for storing and manipulating data can be used. In Section 2 we explain what patterns are, and give briefly some mathematical background for the PBMS. In Section 3 the architecture of the PBMS is described. In Section 4 we list criteria for PBMS characteristics and in Section 5 we conclude.

2. Patterns and Mathematical background

To effectively describe what patterns are in our context and why they are so useful, we present an exemplary scenario, summarized in Figure 1. The available raw data come from a super market, where the goods purchased from each customer are recorded in the database of the super market. While this vast volume of data is not providing us with any particular information on the buying habits of the customers, we can apply a set of knowledge discovery algorithms to the underlying data and come up with some knowledge. In the case of Figure 1, we have applied an association-rules algorithm, thus the knowledge-carrying abstractions of data are association rules. Observe that the association rule of Figure 1, apart from providing the end-user with some hidden knowledge over the underlying data, can be linked to the subset of raw data it is related to (see the arrows in Figure 1).

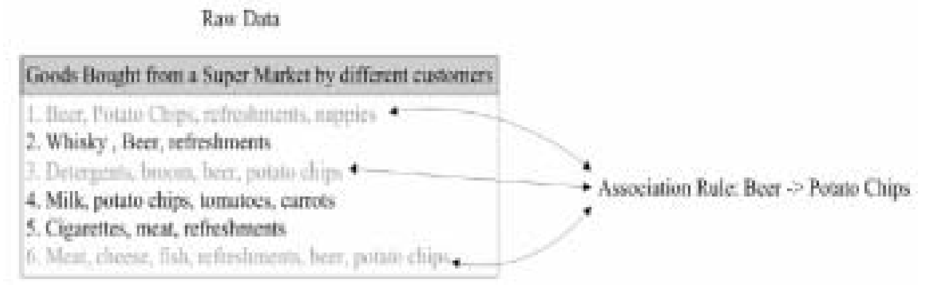

Raw Data
Goods Bought from a Super Market by different customers
1. Beer, Potato Chips, refreshments, nappies
2. Whisky , Beer, refreshments
3. Detergents, broom, beer, potato chips
4. Milk, potato chips, tomatoes, carrots
5. Cigarettes, meat, refreshments
6. Meat, cheese, fish, refreshments, beer, potato chips
Association Rule: Beer -> Potato Chips

Figure 1. Patterns (association rules) and their mapping to raw data [VaHT02]

Patterns, thus, can be regarded as *artifacts*, which describe (a subset of) raw data with similar properties and/or behavior, providing a compact and rich in semantics representation of data. In our setting, patterns must satisfy the following two *properties*:

1. We assume that a mapping between the raw data space and the pattern space is always possible; in the most general case the cardinality of this mapping is `many-to-many`. Hopefully, the amount of data associated to a single pattern through this mapping is large enough to qualify the pattern as a **compact** representation of data.
2. The extracted patterns have to preserve as much knowledge as possible with respect to the raw data they correspond to. In other words, they must be **semantically rich**, both through the knowledge preservation and their conciseness.

Based on the above discussion, we are ready to give a first, informal, definition for patterns.

Definition 1 A pattern can be defined as a compact and rich in semantics representation of raw data. □

Next, we discuss the mathematical foundations of patterns and their relationship to raw data. We can base the discussion on the following assumptions, which directly stem from the aforementioned definitions:

– There exists a *source space* (the space of raw data) and a *pattern space*.
– There always exist *relationships* among the members of the source space and the members of the pattern space. In general, these relationships can be of cardinality `many-to-many`, i.e., a pattern can correspond to more than one data items and vice versa.

Source Space Characteristics. We can assume that each data item in the source space is characterized by a *finite number*, say N, *of dimensions*. In the case of a single relational table, these dimensions would probably correspond to the fields of the table (although one can also consider scenarios where only a subset of these fields is considered, or where extra information, like statistical information derived from catalog histograms, annotates each tuple). We can assume, as usual, that each dimension of the data items belongs to an infinitely countable domain. We will call $dom(x)$ the domain of each dimension. If $A_1,\dots,A_N$ are the dimensions of the data items, the source space is defined as $dom(A_1)x\dots xdom(A_N)$, which we will call D^N. The set of actually stored data comprise a subset of D^N, which we call active source space; we will refer to the active source space as D_A^N .

Pattern Space Characteristics. Independently of which kinds of patterns we are working on, we can assume that *each pattern is characterized by a finite number*, say M, *of dimensions* or *features*. We can assume, as usual, that each dimension of a pattern belongs to an infinitely countable domain. Again, we will call $dom(x)$ the domain of each dimension. If $B_1,\dots,B_M$ are the dimensions of the patterns, we define the pattern space as $dom(B_1)x\dots xdom(B_M)$, which for brevity we will call D^M. Note, that as usually, only a finite subset of the pattern space will be eventually stored, queried and managed inside the pattern management system; similarly to data, we will refer to this subset as the active pattern space D_A^M .

Relationship of data and pattern spaces. Without loss of generality, we can say that each pattern is related to more than one data items. As far as data items are concerned, a data item can also be related to more than one pattern due to:

(a) fuzziness, i.e., the application of an algorithm that does not result in a single crisp pattern value for each data item;

(b) overlapping output, e.g., in the case of a pair of association rules where one is a more general variant of the other;

(c) the simple fact that there can be patterns of different types corresponding to a single data item (e.g., because two different algorithms were executed over the same set of raw data).

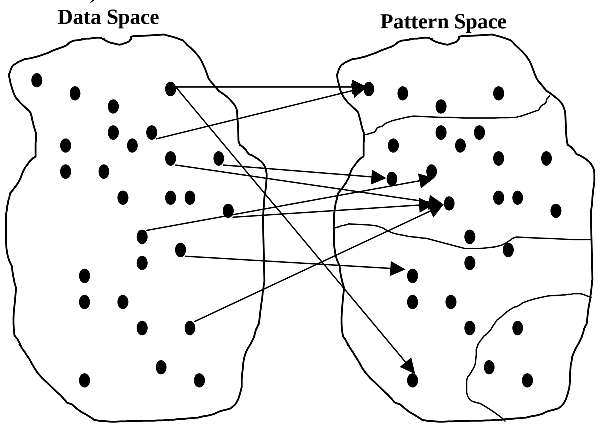


Figure 2. Data and pattern spaces, along with pattern subspaces.

Observe Figure 2, where we can see the relationship between the source space and a certain pattern space. As mentioned before, the relationship is not a function, but in general $f_{DP}:D^N \rightarrow D^M$ is a many-to-many relationship.

Relationship characteristics. By observing the Venn diagram of Figure 2, we can enrich the data-pattern relationship f_{DP} with extra information. Basically, we can define:

(a) *participation measures for the relationship*, e.g., how large subsets of D^N, D^M it involves, if it is total or injective, etc;

(b) *importance measures for a data item*, e.g., how many patterns does it correspond to (practically denoting how important or interesting it might be);

(c) *importance measures for a pattern*, e.g., how many data items it corresponds to, whether the set of its data includes the set of data of another pattern (in which case we can test for pattern zoom-in/out, and of course, identity).

Another issue that we can discuss is that of (a) rich and (b) compact representation of the data items from the part of the patterns. For both of these issues we can give a naive metric. Let us start with richness of representation. Clearly, we cannot always (or, do not wish to) store the entire relationship of patterns and data in persistent storage. Instead, we can introduce *condensed representations* of this relationship (e.g., in the form of expressions in a certain language). The richness of representation can be computed as the fraction of the relationships captured by this condensed representation over the total number of relationships of f_{DP} (actually, we can define two such functions, exactly as the Information Retrieval community defines precision and recall). Another measure could be the success ratio of the 'inverse' mapping, if such a mapping can be derived. As far as the compactness of the representation is concerned, a coarse measure would be the ratio of $size(D_A^M)*M/size(D_A^N)*N$. (Note that M/N is not a good measure, since an association rule over a relation with N attributes can involve up to $2N$ attributes; still if we could characterize 1M data rows with 1K association rules or 5 clusters, this would be compact enough representation).

Subspaces of the pattern space. Another interesting observation lies in Figure 2, where we can see that in general, *the pattern space is the union of a finite set of different sub-spaces*. We can envisage this division in two different cases. In the first case, we can consider a certain subspace comprising the set of association rules over a table of raw data and another subspace comprising the set of decision trees over the same table. In general, exactly as we define tables in the relational paradigm, we can define collections of semantically related patterns (i.e., subspaces of the grand pattern space) at design time. In the second case, one can consider that all the patterns produced by a certain algorithm (e.g., association rules) form a certain subspace (i.e., we have one subspace for decision trees, another for frequent item sets, and so on). Each subspace can be further decomposed in more subspaces, in the same spirit (e.g., the subspace of clusters can be further divided to subspaces according to the type of clustering algorithm / cluster scheme of the patterns). Again, the problem of pattern similarity between the members of the two subspaces is raised. In general, subspaces can be considered as artificial constructs necessary to group semantically and structurally similar patterns together, with the purpose of their further querying.

3. The reference architecture for the Pattern-Base Management System

Patterns can be managed by using a *Pattern-Base Management System* exactly as database records are managed by a database management system. In our setting, a Pattern-Base Management System (PBMS) can be defined as follows.

Definition 2. A **Pattern-Base Management System** (**PBMS**) is a system for handling (storing/processing/retrieving) patterns defined over raw data in order to efficiently support pattern matching and to exploit pattern-related operations generating intentional information. □

The reference architecture of a PBMS is depicted in Figure 3. It consists of three major layers of information organization. In the bottom of Figure 3, we depict the data stores that contain raw data (forming thus, the *Raw Data Layer*). Raw data can be either managed by a DBMS or can be stored in files, streams or any other physical mean that is managed outside a DBMS. At the top of Figure 3, we depict the PBMS repository that contains patterns, which forms the *Pattern-Base Management System Layer*. Finally, in the middle of Figure 3, we can observe the intermediate mappings that relate patterns to their corresponding data, forming the *Intermediate Data Layer*. Intermediate mappings facilitate the justification of any knowledge inferred at the PBMS with respect to the raw data; for example, they could be used to retrieve the rows that produced the association rule of Fig. 1. Ideally, we would like this layer to be part of PBMS, involving specialized storage and indexing structures. For practical purposes, though, the PBMS should be constructed in such a way that it functions even if intermediate results are out of its control (which we would expect as the most possible scenario in real-world scenarios), or even absent.

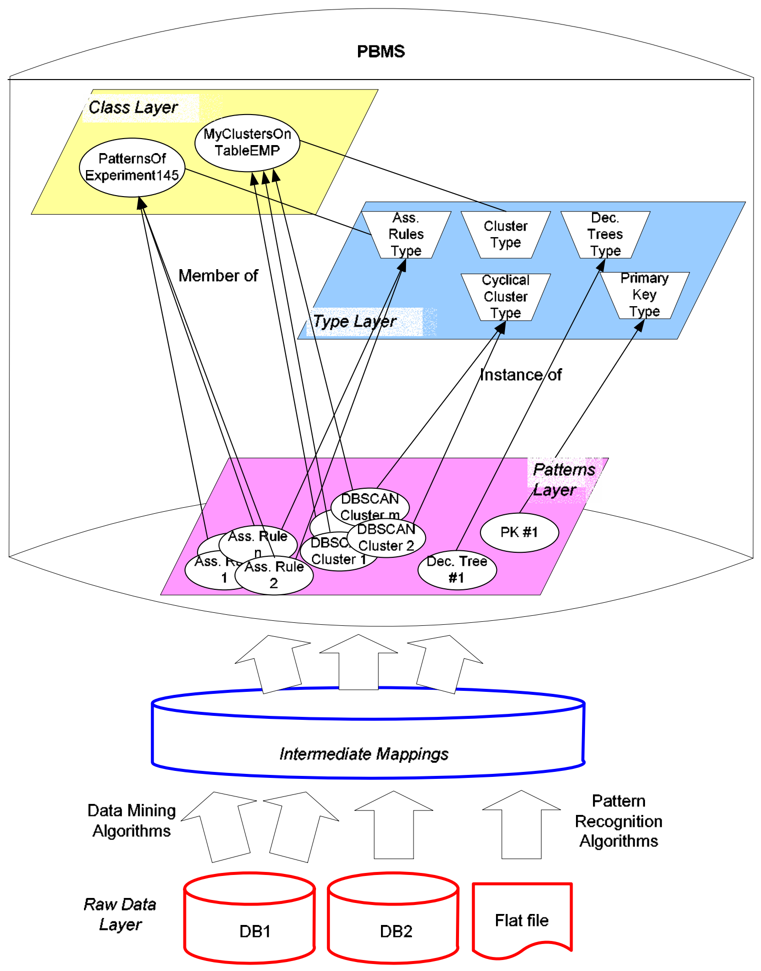


Figure 3. Reference architecture for the Pattern-Base Management System

In the sequel, we will delve into the internals of the PBMS, which will be the focus of our research. We distinguish three internal *conceptual* layers in the architecture of the PBMS (in contrast to the aforementioned *physical* layers of the previous paragraph):

- *Pattern layer*, which is populated with patterns. These are the basic entities that are stored in the PBMS.
- *Type layer*, which holds built-in and user defined types for patterns.
- *Class layer*, which holds built-in and user-defined definitions of pattern classes, i.e., collections of semantically related patterns.

The last two layers are conceptually "higher" than the pattern layer as the entities they define are more abstract. Their role is supportive in the definition and manipulation of the patterns. Next, we present a brief description of all the entities that appear in this conceptual description of the PBMS:

- *Pattern Layer*. The patterns can be either identified through query processing or given a-priori, according to the classification proposed in Section 1. Patterns are related with raw data via the intermediate mappings. In the first case, they can be obtained by different data mining algorithms. In Figure 3, three algorithms have been applied: an algorithm for the extraction of association rules, the DBSCAN algorithm [EKSX996] for the extraction of clusters and an algorithm for the extraction of a decision tree. Besides this, a user can specifically define and store a pattern over the raw data, e.g., a primary key constraint, a functional dependency or an integrity rule. In Figure 3 we depict the case where the designer of the database with the raw data, has defined a primary key constraint over them.
- *Type Layer*. The PBMS pattern types, describe the syntax of the patterns. Patterns of same type share similar characteristics. The intuition behind this is that a large number of operations can be applied to patterns with similar characteristics and get *potentially meaningful* results. Thus, defining a pattern as an instance of a specific type ensures that certain operations will be applicable to it. For example, the overlap of a circle and an inference rule cannot even be defined in an intuitive manner, whereas the overlap between two circles may be of interest. Moreover, the lesson learned from object-oriented databases dictates that different internal representations can be suitable for different types of patterns. Normally, we anticipate the PBMS to come with a set of built-in, popular pattern types. Still, the type layer must be *extensible*, simply because the set of pattern types that it incorporates must be extensible. In fact, we understand the extensibility mechanism as a requirement for the model, so that whenever a new kind of patterns is devised, it can be gracefully incorporated in the PBMS. The type layer must characterize patterns based on the type of source data they are related with, their structure, and the measures we want to associate with them.
- *Class Layer*. The PBMS classes are collections of patterns which share some semantic similarity. Patterns that are members of the same class are obligatorily required to belong to the same type. Classes are used to create patterns with predefined semantics given by the designer; by doing so, the designer makes it easier for the users to work on them in a meaningful way. For example, a class may comprise members that are clusters over `AGE` and `SALARY`, i.e., patterns that represent groups of employees in a company that have approximately the same age and get similar salaries. In order to model these concepts, we assume that supportive entities not appearing in Figure 3, called *features*, belongs to the PBMS. Features form an ontology in the PBMS, whose elements are used in the definitions

of patterns and classes. Features facilitate giving semantics to the patterns or classes that are defined over them. We can imagine features as a dynamic interface over raw data. For example, a feature labeled `AGE` can be dynamically linked to several data representing the age of employees.

One could elaborate more on extra potential characteristics of the class and type layers; in fact, in [Riz+03] we have introduced the notion of specialization hierarchies among types and the possibility of composing/refining composite types. For lack of space, we refer the interested reader to [Riz+03].

Another meta-entity of the PBMS is *the language* that is used to define the structure of patterns and the condensed expression for representing the mappings between the patterns and their respective raw data. The expressive power of the language and the available features determine the kind of patterns that we can express in a certain PBMS. The PBMS features provide the *semantic domain* of the PBMS; the supported language on the other hand, provides the *expressive power* of the PBMS. By extending the defined features and/or by using a more expressive language, more patterns can be stored and manipulated by the PBMS.

5. Towards a PBMS Manifesto

In this section we will discuss on the necessary principles and features for a PBMS. Clearly, this represents a personal viewpoint; still we believe it is well justified by interesting practical applications. First, we give a short list of requirements and characteristics for a PBMS. We group the requirements in functional and system areas. Then, we will briefly compare our approach with respect to the current state of the art in object-relational and object-oriented DBMS's.

Data model

1. A PBMS will be based upon a *generic uniform model* that covers all kinds of patterns. The model captures all static, dynamic and well-formedness properties of *all* patterns (i.e., structure, operations and integrity constraints), just as the relational model does for database records.
2. A PBMS model represents its instances in a *compact* and *rich in semantics* fashion (i.e., we gain a concise representation with respect to the raw data and we do not lose information; in fact, we abstract new knowledge by escaping from the vast volume of raw data).
3. A PBMS will support different types of patterns in an *extensible fashion*. Whenever new kinds of patterns are discovered, or considered to be of interest to the PBMS users, they will be smoothly integrated in the PBMS, through an extensibility mechanism.
4. A PBMS will allow its user to be able to identify interesting subsets of the pattern space on the basis of their *semantical similarity*. This corresponds to the aforementioned notion of classes.
5. A PBMS will support *composite* patterns, generated from simpler ones. This implies that several levels of representation/abstraction should exist among

patterns, including different levels of granularity, multi-dimensionality, recursion, hierarchies, etc.

Architecture

6. A PBMS will have its own mechanisms for *representing and storing its entries*.
7. A PBMS will cooperate with DBMS's storing *raw data*; however raw data can also be stored outside a DBMS.
8. A PBMS must be able to manage also *pattern extraction and creation*, through the appropriate mechanisms.
9. A PBMS will allow access to *intermediate results of pattern creation algorithms*; still, whenever access to these data is not possible, the PBMS will continue to function properly, compensating the loss of knowledge through appropriate mechanisms.

Query Language and Processing

10. A PBMS processes different kinds of queries (because of different user needs), possibly even on raw data and *returns more intuitive results to users*.
11. A PBMS employs a query language which can at least perform the following tasks:
    - *Pattern matching*, meaning the mapping of a certain (new) input to a set of patterns already found in the PBMS and its reverse operation, which leads from the pattern towards the raw data that correspond to it
    - *Deductive capabilities*, meaning that logical inferences on the basis of the pattern representation are one of the desiderata for the query language. This can involve several operations, such as *zoom in/out* over patterns that can be composed with part-of relationships (i.e., a pattern that corresponds to a large set of data is possibly related to a pattern that corresponds to a subset of these data), *pattern composition* of patterns, which are candidates to compose a complex pattern, *roll-up and drill-down* over multidimensional hierarchies of features, etc.
    - *Meta querying facilities*, enabling the possibility of querying the composing elements of a pattern base (e.g., types, classes, etc.) in order to extract meaningful conclusions and to enable the administration of the PBMS
12. A PBMS is useful in order to process those queries *more efficiently than a normal DBMS* would do.

We tend to believe that the PANDA approach is not yet another object-oriented or object-relational variant. Although it might look quite close due to the class/type system, the two approaches are intrinsically different:

- In terms of the model, the discriminating feature is the requirement for a semantically rich representation of patterns, achieved by separating structure and measure on the one hand, by introducing the expression component on the other.
- From the functional point of view, instead of pointer-chasing object-oriented queries, novel querying requirements arise, including (a) ad hoc operations over the source and pattern spaces and their mapping, (b) pattern matching tests,

facilitated from the separation of structure and measure, and (c) reasoning facilities based on the expression component of patterns.

– In terms of system architecture, the relevance of queries aimed at evaluating similarity between patterns and the request for efficiency call for alternative storage and query optimization techniques.

6. Conclusions and Future Work

In this paper, we have dealt with the introduction of the mathematical foundations and the architecture for pattern management. First, we introduced the intuition and mathematical foundations for pattern management. Next, we presented the architecture of a *Pattern Base Management System* that can be used to efficiently store and query patterns. Finally, we commented upon the criteria and characteristics of PBMS's and on whether the existing systems for storing and manipulating data can cover the special user requirements that patterns impose.

Clearly, under this general framework, the PANDA consortium has several interesting tasks to accomplish. From our part, our future research includes primarily theoretical aspects like the introduction of a powerful language for expressing pattern semantics, the possibility of contributing to the model of [Riz+03] e.g., by introducing constraints on patterns; and finally by contributing to a flexible query language and its respective algebra for retrieving and comparing complex patterns.

References

[BeLi96] Michael Berry, Gordon Linoff. "Data Mining Techniques: For Marketing, Sales, and Customer Support". John Willey, 1996.

[EKSX96] M. Ester, H.-P. Kriegel, J. Sander, X. Xu. A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. Proceedings of the Second International Conference on Knowledge Discovery and Data Mining (KDD-96): pp. 226-231, 1996

[FPSU96] U. Fayyad, G. Piatesky-Shapiro, P. Smuth & R. Uthurusamy (eds). "From Data Mining to Knowledge Discovery: An Overview". Advances in Knowledge Discovery and Data Mining. AAAI Press, 1996.

[HaKa01] Jiawei Han, Micheline Kamber. "Data Mining: Concepts and Techniques". Academic Press, 2001

[Riz+03] S. Rizzi, E. Bertino, B. Catania, M. Golfarelli, M. Halkidi, M. Terrovitis, P. Vassiliadis, M. Vazirgiannis, E. Vrachnos. Towards a logical model for patterns. *Submitted for publication.*

[VaHT02] M. Vazirgiannis, M. Halkidi, G. Tsatsaronis. The logical model of a Pattern Management System (The AUEB viewpoint). PANDA Internal Report PANDA -- AUEB – 001. June 2002.

[Vaz+02] M. Vazirgiannis, M. Halkidi, G. Tsatsaronis, E. Vraxnos, D. Keim, P. Xeros, Y. Panagis, A. Pikrakis (UoA). Related Research. PANDA Internal Report, PANDA – AUEB, KONSTANZ, UoA, CTI-- <001> . October 2002.

A Logical Framework for Pattern Representation

Stefano Rizzi[1], Elisa Bertino[2], Barbara Catania[3], and Matteo Golfarelli[1]

[1] DEIS, Univ. of Bologna, Italy
[2] DICO, Univ. of Milan
[3] DISI, Univ. of Genoa, Italy

Abstract. Nowadays, the vast volume of collected digital data obliges us to employ processing methods like pattern recognition and data mining in order to reduce the complexity of data management. The output of these techniques are heterogeneous knowledge artifacts, compact and rich in semantics, that we call *patterns*. Patterns can be efficiently stored and managed by using a Pattern-Based Management System. Inside such a system, patterns have to be represented and manipulated according to a *pattern logical model*. In this paper we present the logical foundations of such a model, introducing the notions of pattern types and pattern classes, which stand for the intensional and extensional description of pattern instances, respectively. The framework is general and extensible enough to cover a broad range of real-world patterns; each pattern is characterized by its structure, the related underlying data, an expression that carries the semantics of the pattern, and measurements of how successful the representation of raw data is. Some remarkable types of relationships between patterns are also discussed.

1 Introduction

The increasing opportunity of quickly collecting and cheaply storing large volumes of data, and the need for extracting concise information to be efficiently manipulated and intuitively analysed, are posing new requirements for DBMSs in both industrial and scientific applications. In this context, data warehousing systems have been defined with the aim of providing managers and knowledge workers with ready-at-hand summary data to be used for decision support. However, the limited analysis power provided by OLAP interfaces proved to be insufficient for advanced applications, in which the huge quantity of data stored necessarily requires semi-automated processing techniques, and the peculiarity of the user requirements calls for non-standard analysis techniques. Thus, sophisticated data processing tools (based for instance on data mining, pattern recognition, and knowledge extraction techniques) were devised in order to reduce, as far as possible, the user intervention in the process of extracting interesting knowledge artifacts (e.g., clusters, association rules, time series) from raw data [6, 7, 9]. We claim that the term *pattern* is a good candidate to generally denote these novel information types, characterized by a high degree of diversity and complexity.

Differently from the case of data warehousing systems, the problem of directly storing and querying pattern-bases has received very limited attention so far in the commercial world, and probably no attention at all from the database community. Indeed, traditional DBMSs do not seem to be powerful and flexible enough to deal with this new kind of knowledge. On the other hand, we claim that pattern representation and management can be efficiently provided by using a specific system, called *Pattern-Base Management System* (PBMS), capable of modeling, storing, and efficiently managing patterns. The advantages in using a PBMS can be summarized as follows:

- *Abstraction.* Within a PBMS, patterns would be made first-class citizens thus providing the user with a meaningful abstraction of raw data to be directly analyzed and manipulated.
- *Efficiency.* Introducing an architectural separation between the PBMS and the DBMS would improve the efficiency of both traditional transactions on the DBMS and advanced processing on patterns.
- *Querying.* The PBMS would provide an expressive language for querying the pattern-base in order to retrieve and compare patterns.

Similarly to any Database Management System, also a PBMS must rely on a specific data model, by which patterns can be represented and manipulated. Such a model must satisfy the following requirements:

- *Generality.* The model must be general enough to meet all the specific requirements posed in different application domains for different kinds of patterns.
- *Extensibility.* The model must be extensible to accomodate new kinds of patterns introduced by novel and challenging applications.
- *Reusability.* The model must include constructs encouraging the reuse of what has already been defined.

In this paper, we present a logical framework for pattern representation, which represents the first step towards the definition of a pattern data model. Informally, a pattern can be thought of as a *compact* and *rich in semantics* representation of raw data. In our approach, in order to guarantee generality, a pattern is modeled by some main components, namely its *structure*, *measure*, *source*, and *expression*. The structure component *qualifies* the pattern by locating it within a pattern space. The measure component *quantifies* the pattern by measuring the quality of the raw data representation achieved by the pattern itself. The source component describes the raw data the pattern relates to. The expression component describes the (approximate) mapping between the raw data space and the pattern space. We claim that such components are sufficient to represent a large number of patterns, arising in different application domains.

Each pattern is an instance of a *pattern type*. Instances of the same type share similar characteristics. Thus, defining a pattern as an instance of a specific type ensures that certain operations can be applied to it. Moreover, each pattern can belong to one or more classes. Each class represents a collection of patterns with the same type and it is the key concept for the definition of a query language since, similarly to the object-oriented context, each query is executed against one or more collections.

In order to guarantee extensibility and reusability, we then propose several interesting relationships between pattern types, such as specialization, composition, and refinement.

It is important to note that, even if some specific approaches have already been proposed to model patterns, such as Predictive Model Markup Language [4], SQL/MM standard [2], Common Warehouse Model [1], Java Data Mining API [3], they seem inadequate to represent and handle different classes of patterns in a flexible, effective, and coherent way. In fact, they lack the requirements listed above since usually, only a given list of predefined pattern types can be represented and no general and extensible approach to pattern modeling is proposed. Also the direct usage of a more general and well-understood technology, such as the object-oriented or object-relational one, does not seem to solve the problem, since pattern modeling and generic object modeling are intrinsincally different. The discriminating feature is the requirement for a semantically rich representation of patterns, achieved by separating structure and measure on the one hand, by introducing the expression component on the other. These components have a well defined semantics and a PBMS relying on the proposed model will be able to manage all of them in a semantically meaningful way.

The paper is structured as follows. In Section 2, we introduce the concept of pattern type. Patterns are introduced in Section 3 whereas classes are presented in Section 4. In Section 5 we discuss some remarkable types of relationships between patterns. Finally, in Section 6 we present some conclusions and outline future work.

2 Pattern types

Though our approach is parametric on the typing system adopted, the examples provided in this paper will be based on a specific, very common typing system. Assuming there is a set of *base types* (including the *root type* $\perp$) and a set of *type constructors*, the set T of types includes all the base types together with all the types recursively defined by applying a type constructor to one or more other types. Types are applied to *attributes*.

Let base types include integers, reals, Booleans, strings, and timestamps; let type constructors include list, set, bag, array, and tuple. Using an obvious syntax, some examples of type declarations are (we use uppercase for base types and type constructors, lowercase for attributes):

- salary: REAL
- SET(INTEGER)
- TUPLE(x: INTEGER, y: INTEGER)
- personnel: LIST(TUPLE(age: INTEGER, salary: INTEGER))

A pattern type represents the intensional form of patterns, giving a formal description of their structure and relationship with source data. Thus, pattern types play the same role of abstract data types in the object-oriented model.

Definition 1 (Pattern type). *A* pattern type *pt is a quintuple $pt = (n, ss, ds, ms, f)$ where n is the* name *of the pattern type; ss, ds, and ms (called respectively* structure schema*,* source schema*, and* measure schema*) are types in T; f is a* formula*, written in a given language, which refers to attributes appearing in the source and in the structure schemas.*

The first component of a pattern type has an obvious meaning; the remaining four have the following roles:

- The structure schema ss defines the pattern space by describing the structure of the patterns instances of the pattern type. The achievable complexity of the pattern space (hence, the flexibility of pattern representation) depends on the expressivity of the typing system.
- The source schema ds defines the related source space by describing the dataset from which patterns, instances of the pattern type being defined, are constructed. Characterizing the source schema is fundamental for every operation which involves both the pattern space and the source space (e.g., when applying some technique to extract patterns from raw data or when checking for the validity of a pattern on a dataset).
- The measure schema ms describes the measures which quantify the quality of the source data representation achieved by the pattern. The role of this component is to enable the user to evaluate how accurate and significant for a given application each pattern is. Besides, the different semantics of the measure component with reference to the structure can be exploited in order to define more effective functions for evaluating the distance between two patterns [8].
- The formula f describes the relationship between the source space and the pattern space, thus carrying the semantics of the pattern. Inside f, attributes are interpreted as free variables ranging over the components of either the source or the pattern space. Note that, though in some particular domains f may exactly express the inter-space relationship (at most, by allowing all raw data related to the pattern to be enumerated), in most cases it will describe it only approximatively.

Though our approach to pattern modeling is parametric on the language adopted for formulas, the achievable semantics for patterns strongly depends on its expressivity. For the examples reported in this paper, we try a constraint calculus based on polynomial constraints which seems suitable for several types of patterns [10]; still, a full exploration of the most suitable language is outside the scope of the paper.

Example 1. Given a domain D of values and a set of transactions, each including a subset of D, an *association rule* takes the form $A \rightarrow B$ where $A \subset D$, $B \subset D$, $A \cap B = \emptyset$. A is often called the *head* of the rule, while B is its *body* [9]. An example of a pattern type for modeling association rules over strings representing products is the following:

$$\begin{aligned}
n &: \mathsf{AssociationRule}\\
ss &: \mathsf{TUPLE(head{:}\ SET(STRING),\ body{:}\ SET(STRING))}\\
ds &: \mathsf{BAG(transaction{:}\ SET(STRING))}\\
ms &: \mathsf{TUPLE(confidence{:}\ REAL,\ support{:}\ REAL)}\\
f &: \forall x (x \in \mathsf{head} \vee x \in \mathsf{body} \Rightarrow x \in \mathsf{transaction})
\end{aligned}$$

The structure schema is a tuple modeling the head and the body. The source schema specifies that association rules are constructed from a bag of transactions, each defined as a set of products. The measure schema includes two common measures used to assess the relevance of a rule: its confidence (what percentage of the transactions including the head also include the body) and its support (what percentage of the whole set of transactions include both the head and the body). Finally, the formula of the constraint calculus represents (exactly, in this case) the pattern/dataset relationship by associating each rule with the set of transactions which support it.

Example 2. An example of a mathematical pattern is a straight line which interpolates a set of samples. In this case, the source schema models the samples, the structure schema includes the two coefficients necessary to determine a line, while the measure schema includes, for instance, a fitting quantifier. The formula which establishes the approximate correspondence between the pattern and the source data is the equation of the line.

$$\begin{aligned} n &: \mathsf{InterpolatingLine} \\ ss &: \mathsf{TUPLE(a{:}\ REAL,\ b{:}REAL)} \\ ds &: \mathsf{SET(sample{:}\ TUPLE(x{:}\ REAL,\ y{:}\ REAL))} \\ ms &: \mathsf{fitting{:}\ REAL} \\ f &: \mathsf{y = a \cdot x + b} \end{aligned}$$

3 Patterns

Before introducing the concept of pattern, we define pattern values. To this purpose, we assume that each base type t_i is associated with a set of known values $dom(t_i)$. Values for any other type t are inductively defined as follows: (i) if t =SET(t_1) and $v_1, \ldots, v_n$ are values for t_1, then $\{v_1, \ldots, v_n\}$ is a value for t; (ii) if t =BAG(t_1) and $v_1, \ldots, v_n$ are values for t_1, then $\{\{v_1, \ldots, v_n\}\}$ is a value for t; (iii) if t =LIST(t_1) and $v_1, \ldots, v_n$ are values for t_1, then $< v_1, \ldots, v_n <$ is a value for t; (iv) if t =ARRAY$[1, \ldots, n](t_1)$ and $v_1, \ldots, v_n$ are values for t_1, then $[v_1, \ldots, v_n]$ is a value for t; (v) if t =TUPLE($a_1 : t_1, \ldots, a_n : t_n$) and $v_1, , v_n$ are values for $t_1, \ldots, t_n$, respectively, then $(a_1 = v_1, \ldots, a_n = v_n)$ is a value for t.

Let raw data be stored in a number of databases and/or files. A *dataset* is any subset of these data, which we assume to be wrapped under a type of our typing system (*dataset type*). Patterns can be defined as follows.

Definition 2 (Pattern). *Let $pt = (n, ss, ds, ms, f)$ be a pattern type. A* pattern *p instance of pt is a quintuple $p = (pid, s, d, m, e)$ where pid (*pattern identifier*) is a unique identifier for p; s (*structure*) is a value for type ss; d (*source*) is a dataset whose type conforms to type ds; m (*measure*) is a value for type ms; e is an expression denoting the region of the source space which is related to pattern p.*

According to this definition, a pattern is characterized by (1) a pattern identifier (which plays the same role of OIDs in the object model), (2) a structure that positions the pattern within the pattern space defined by its pattern type, (3) a source that identifies the specific dataset the pattern relates to, (4) a measure that estimates the quality of the raw data representation achieved by the pattern, (5) an expression which relates the pattern to the source data. In particular, the expression is obtained by the formula f in the pattern type by (1) instantiating each attribute appearing in ss with the corresponding value specified in s, and (2) letting the attributes appearing in ds range over the source space. Note that further information could be associated to each pattern, specifying for instance the mining session which produced it, which algorithm was used, which parameter values rule the algorithm, etc.

Example 3. Consider again pattern type AssociationRule defined in Example 1, and suppose that raw data include a relational database containing a table sales which stores data related to the sales transactions in a sport shop: sales (<u>transactionId</u>, article, quantity). Using an extended SQL syntax to denote the dataset, an example of an instance of AssociationRule is:

$$
\begin{aligned}
pid &: 512\\
s &: (\mathsf{head} = \{\text{'Boots'}\}, \mathsf{body} = \{\text{'Socks'}, \text{'Hat'}\})\\
d &: \text{'SELECT SETOF(article) AS transaction}\\
&\quad \text{FROM sales GROUP BY transactionId'}\\
m &: (\mathsf{confidence} = 0.75, \mathsf{support} = 0.55)\\
e &: \{\mathsf{transaction} : \forall x(x \in \{\text{'Boots'}, \text{'Socks'}, \text{'Hat'}\} \Rightarrow x \in \mathsf{transaction})\}
\end{aligned}
$$

In the expression, transaction ranges over the source space; the values given to head and body within the structure are used to bind variables head and body in the formula of pattern type AssociationRule.

Example 4. Let raw data be an array of real values corresponding to samples periodically taken from a signal, and let each pattern represent a recurrent waveshape together with the position where it appears within the dataset and its amplitude shift and gain:

$$
\begin{aligned}
n &: \mathsf{TimeSeries}\\
ss &: \mathsf{TUPLE(curve{:}\ ARRAY[1..5](REAL),\ position{:}\ INTEGER,}\\
&\quad \mathsf{shift{:}\ REAL,\ gain{:}\ REAL)}\\
ds &: \mathsf{samples{:}\ ARRAY[1..100](REAL)}\\
ms &: \mathsf{similarity{:}\ REAL}\\
f &: \mathsf{samples[position} + i - 1] = \mathsf{shift} + \mathsf{gain} \times \mathsf{curve}[i],\\
&\quad \forall i : 1 \le i \le 5
\end{aligned}
$$

Measure similarity expresses how well waveshape curve approximates the source signal in that position. The formula approximatively maps curve onto the data space in position. A possible pattern, extracted from a dataset which records the hourly-detected levels of the Colorado river, is as follows:

$$
\begin{aligned}
pid &: 456\\
s &: (\mathsf{curve} = (\mathsf{y} = 0, \mathsf{y} = 0.8, \mathsf{y} = 1, \mathsf{y} = 0.8, \mathsf{y} = 0),\\
&\quad \mathsf{position} = 12, \mathsf{shift} = 2.0, \mathsf{gain} = 1.5)\\
d &: \text{'colorado.txt'}\\
m &: \mathsf{similarity} = 0.83\\
e &: \{\mathsf{samples}[12] = 2.0, \mathsf{samples}[13] = 3.2, \mathsf{samples}[14] = 3.5,\\
&\quad \mathsf{samples}[15] = 3.2, \mathsf{samples}[16] = 2.0\}
\end{aligned}
$$

4 Classes

A class is a set of semantically related patterns and constitutes the key concept in defining a pattern query language. A class is defined for a given pattern type and contains only patterns of that type. Moreover, each pattern must belong to at least one class. Formally, a class is defined as follows.

Definition 3 (Class). *A class c is a triple c = (cid, pt, pc) where cid (*class identifier*) is a unique identifier for c, pt is a pattern type, and pc is a collection of patterns of type pt.*

Example 5. The *Apriori* algorithm described in [5] could be used to generate relevant association rules from the dataset presented in Example 3. All the generated patterns could be inserted in a class called *SaleRules* for pattern type AssociationRule defined in Example 1. The collection of patterns associated with the class can be later extended to include also rules generated from a different dataset, representing for instance the sales transaction recorded in a different store.

5 Relationships between patterns

In this section we informally introduce some interesting relationships between patterns aimed at increasing the modeling expressivity of our logical framework, but which also improve reusability and extensibility and impact the querying flexibility.

5.1 Specialization

Abstraction by specialization (the so-called *IS-A* relationship) is widely used in most modeling approaches, and the associated inheritance mechanism significatively addresses the extensibility and reusability issues by allowing new entities to be cheaply derived from existing ones.

Specialization between pattern types can be defined by first introducing a standard notion of subtyping between base types (e.g., integer is a subtype of real). Subtyping can then be inductively extended to deal with types containing type constructors: t_1 specializes t_2 if the outermost constructors in t_1 and t_2 coincide and each component in t_2 is specialized by one component in t_1. Finally, pattern type pt_1 specializes pattern type pt_2 (thus pt_1 is a sub-pattern type of pt_2) if the structure schema, the source schema, and the measure schema of pt_1 specialize the structure schema, the source schema, and the measure schema of pt_2.

Note that, if pt_1 specializes pt_2 and class c is defined for pt_2, also the instances of pt_1 can be part of c.

Example 6. Given a set S of points, a clustering is a set of clusters, each being a subset of S, such that the points in a cluster are more similar to each other than points in different clusters [9]. The components for pattern type Cluster, representing circular clusters defined on a 2-dimensional space, are:

$$\begin{aligned} n &: \mathsf{Cluster} \\ ss &: \mathsf{TUPLE(radius{:}\ \bot,\ center{:}\ TUPLE(cx{:}\ \bot,\ cy{:}\ \bot))} \\ ds &: \mathsf{SET(x{:}\ \bot,\ y{:}\ \bot)} \\ ms &: \emptyset \\ f &: (\mathsf{x}-\mathsf{cx})^2 + (\mathsf{y}-\mathsf{cy})^2 \leq \mathsf{radius}^2 \end{aligned}$$

where f gives an approximate evaluation of the region of the source space represented by each cluster. While in Cluster the 2-dimensional source schema is generically defined, clusters on any specific source space will be easily defined by specialization. Thus, Cluster could be for instance specialized into a new pattern type ClusterOfIntegers where cx, cy, x, and y are specialized to integers, radius is specialized to reals, and a new measure avgIntraClusterDistance of type real is added, obtaining the following pattern type:

$$\begin{aligned} n &: \mathsf{ClusterOfInteger} \\ ss &: \mathsf{TUPLE(radius{:}\ REAL,} \\ &\quad\ \mathsf{center{:}\ TUPLE(cx{:}\ INTEGER,\ cy{:}\ INTEGER))} \\ ds &: \mathsf{SET(x{:}\ INTEGER,\ y{:}\ INTEGER)} \\ ms &: \mathsf{avgIntraClusterDistance{:}\ REAL} \\ f &: (\mathsf{x}-\mathsf{cx})^2 + (\mathsf{y}-\mathsf{cy})^2 \leq \mathsf{radius}^2 \end{aligned}$$

5.2 Composition and refinement

A nice feature of the object model is the possibility of creating complex objects, i.e. objects which consist of other objects. In our pattern framework, this can be achieved by extending the set of base types with pattern types, thus giving the user the possibility of declaring *complex types*. This technique may have two different impacts on modeling.

Firstly, it is possible to declare the structure schema as a complex type, in order to create patterns recursively containing other patterns thus defining, from the conceptual point of view, a *part-of* hierarchy. We will call *composition* this relationship.

Secondly, a complex type may appear within the source schema: this allows for supporting the modeling of patterns obtained by mining other existing patterns. Since in general a pattern is a compact representation of its source data, we may call *refinement* this relationship in order to emphasize that moving from a pattern type to the pattern type(s) that appear in its source entails increasing the level of detail in representing knowledge.

Example 7. Let pattern type ClusterOfRules describe a mono-dimensional cluster of association rules: the source schema here represents the space of association rules, and the structure models one cluster-representative rule. Assuming that each cluster trivially includes all the rules sharing the same head, it is modeled as follows:

$$\begin{aligned} n &: \mathsf{ClusterOfRules} \\ ss &: \mathsf{representative\colon\ AssociationRule} \\ ds &: \mathsf{SET(rule\colon\ AssociationRule)} \\ ms &: \mathsf{TUPLE(deviationOnConfidence\colon\ REAL,\ deviationOnSupport\colon\ REAL)} \\ f &: \mathsf{rule.ss.head} = \mathsf{representative.ss.head} \end{aligned}$$

where a standard dot notation is adopted to address the components of pattern types. Thus, there is a refinement relationship between ClusterOfRules and AssociationRule.

Consider now that a clustering is a set of clusters; intuitively, also clustering is a pattern, whose structure is modeled by a complex type which aggregates a set of clusters and can be defined as follows:

$$\begin{aligned} n &: \mathsf{Clustering} \\ ss &: \mathsf{clusters\colon\ SET(ClusterOfRules)} \\ ds &: \mathsf{SET(rule\colon\ AssociationRule)} \\ ms &: \mathsf{clusteringValidity\colon\ REAL} \\ f &: \bigvee_{c \in \mathsf{clusters}} c.f \end{aligned}$$

In this case, a composition relationship exists between pattern types Clustering and ClusterOfRules.

5.3 Dependency

When a source schema is declared within a pattern type, two aspects are specified: the *type* of the datasets related to patterns (which defines the "shape" of the source space), and a set of *names* carrying some universal meaning within the application domain (which assigns some semantics to the source space). We call *features* these names.

We observe that tightly binding a pattern type to one application domain by specifying the features within the source schema is unpractical, since it requires to define a different pattern type for each different application domain even if the structure schema, the measure schema, and the expression template are shared. On the other hand, when the end-user manipulates a pattern, it is fundamental for her to know the semantics of the raw data that pattern is related to, particularly when pattern comparisons are to be made.

To overcome this problem we introduce *template types*. A template type is a pattern type in which the source schema includes one or more *general names*, i.e. type names which carry no semantics and merely act as placeholders. Before a pattern can be created, the template must be bound by associating each general name with a feature of the application domain.

From a conceptual point of view, this corresponds to recognizing that, besides the source and the pattern spaces, there may also be a separate *feature space* which acts as a bridge between them by binding the source space to the semantics of a given domain. In modeling terms, this could be represented by a *dependency* relationship (the name is borrowed by UML, where dependencies with ≪binds≫ stereotype are used to instantiate templates).

Example 8. Consider the pattern type ClusterOfInteger, presented in Example 6, where the coordinates of the source schema are x and y and the general names for the structure schema are radius, center, cx, and cy. Semantically rich features, such as Age, Salary, can be associated with the general names x and y respectively, when patterns are generated. The following is an example of a pattern obtained in this way:

$$\begin{aligned}
pid &: 234\\
s &: \text{RECORD(radius: 2.3, center: RECORD(cx: 35, cy: 1500))}\\
d &: \text{SELECT Year(Today())-Year(E.DateOfBirth) AS Age, E.Salary AS Salary}\\
&\quad \text{FROM Employee AS E}\\
m &: \text{Average Intra Cluster distance} : 1.1\\
e &: (\text{Age} - 35)^2 + (\text{Salary} - 1500)^2 \leq 2.3^2
\end{aligned}$$

6 Conclusions and future work

In this paper, we presented a logical framework for pattern representation. Such a framework is the basis for the development of a PBMS. The proposed framework is based on the principles of generality, extensibility and reusability. To address the generality goal we introduced a simple yet powerful modeling framework, able to cover a broad range of real-world patterns. Thus, the most general definition of a pattern specifies its structure, the underlying data that correspond to it, an expression which is rich in semantics so as to characterize what the pattern stands for, and measurements of how successful the raw data abstraction is. To address the extensibility and usability goals, we introduced type hierarchies, which provide the PBMS with the flexibility of smoothly incorporating novel pattern types, as well as mechanisms for constructing composite patterns and for refining patterns.

Though the fundamentals of pattern modeling have been addressed, several important issues still need be investigated. Future research includes both theoretical aspects as well as implementation-specific issues. The theoretical aspects include the evaluation and comparison of the expressivity of different languages to express formulas, and the study of a flexible query language for retrieving and comparing complex patterns. Implementation issues involve primarily the construction of ad-hoc storage management and query processing modules for the efficient management of patterns.

References

1. Common Warehouse Metamodel (CWM). `http://www.omg.org/cwm`, 2001.
2. ISO SQL/MM Part 6. `http://www.sql-99.org/SC32/WG4/Progression_Documents/FCD/fcd-datamining-2001-05.pdf`, 2001.
3. Java Data Mining API. `http://www.jcp.org/jsr/detail/73.prt`, 2003.
4. Predictive Model Markup Language (PMML). `http://www.dmg.org/pmmlspecs_v2/pmml_v2_0.html`, 2003.
5. R. Agrawal and R. Srikant. Fast algorithms for mining association rules. In *Proc. 20th VLDB*, 1994.
6. M. Berry and G. Linoff. *Data mining techniques: for marketing, sales, and customer support.* John Wiley, 1996.
7. U. Fayyad, G. Piatesky-Shapiro, and P. Smyth. From data mining to knowledge discovery: an overview. In *Advances in Knowledge Discovery and Data Mining*, pages 1–34. AAAI Press and the MIT Press, 1996.
8. V. Ganti, R. Ramakrishnan, J. Gehrke, and W.-Y. Loh. A framework for measuring distances in data characteristics. *PODS*, 1999.
9. J. Han and M. Kamber. *Data mining: concepts and techniques.* Academic Press, 2001.
10. P. Kanellakis, G. Kuper, and P. Revesz. Constraint Query Languages. *Journal of Computer and System Sciences*, 51(1):25–52, 1995.

Patterns in hypermedia

Michalis Vaitis[1], Manolis Tzagarakis[2, 3] and George Gkotsis[2, 3]

[1] Department of Geography, University of the Aegean, Greece
[2] Computer Technology Institute, Patras, Greece
[3] Department of Computer Engineering and Informatics, University of Patras, Greece
vaitis@aegean.gr, tzagara@cti.gr, gkotsis@ceid.upatras.gr

Introduction

According to [18], *patterns* are knowledge artifacts that represent properties, rules or taxonomies that hold among data elements. Usually, these artifacts are inferred from the data using data mining techniques and are continuously tested for their validity as the data set evolves. There is a clear distinction between the data elements of a dataset and the pattern (or patterns) that are effective among them at a certain point in time.

Phrased differently, patterns can be regarded as exhibiting *primacy of structure over data*. One consequence of such shift is the elevation of *relationships* from *second class* to a *first class* abstraction within the system.

In recent years, another research area that is involved in the management of relationships among data elements is *structural computing*. As its name reveals, structural computing focuses on the structural abstractions that exist among data in different application domains. In structural computing, data is regarded as an *outcome* of structure. Driven by the philosophy of the "primacy of structure over data" [8], *structural elements* are introduced and treated as first-class entities. *Computations over structure* are defined in order to model the behavioral aspects of the application domain. Computations contribute to the semantic aspects of structure elements. A crucial aspect of structural computing is the fact, that there is no "minimal data abstraction", such as the *relation* in RDBMSs or the *object* in OODBMS. Structural computing introduces the *relationship* as its fundamental building block, allowing the *granularity level* of such systems to be configurable. The issue of granularity seems to be a critical characteristic for structure-oriented approaches.

In this paper we present patterns that appeared in the hypermedia field and outline how these have been represented in the Callimachus hypermedia system [15]. Callimachus is a hypermedia system based on the concepts of structural computing where hypermedia domain aspects are represented as patterns (*hypermedia patterns*). The insights of how such hypermedia patterns are treated in Callimachus may reveal additional requirements on the modeling of patterns in Pattern Based Management Systems (PBMS).

Hypermedia domains: Patterns of information organization

A hypermedia domain is a coherent set of abstractions, that are appropriate to solve organization problems in a particular application domain. These abstractions originated from the classic node-anchor-link model, but mutated in order to fit better entities and relationships present in the application domain. Below we briefly present some of these domains, highlighting representative abstractions and behaviors. Table 1 summarizes the presentation of the hypertext domains.

Navigational hypertext

The navigational domain was one of the first domains supported in early hypertext systems, in which the ability to associate was explicitly provided [7,14].

Inspired by Bush's ideas [1], many researchers have investigated navigational hypertext in computer applications, resulting in many different hypertext systems. As stated in the Dexter Open Hypermedia work [2], the basic node/link network structure is the essence of (navigational) hypertext. As a result, the navigational hypertext can be viewed as a network of data containing 'components' interconnected by relational 'links'. To support the navigational domain, many hypermedia researchers have adopted a fundamental entity that provides the node, link, anchor and context abstractions [10]. A *node* provides a wrapper for an arbitrary resource and may have several anchors associated with it, whereas an *anchor* can be bound to several links. A *link* of the navigational domain is defined as an association between anchors with its direction clearly stated [2] and a *context* is a collection of object references (similar to hypertexts in Dexter and to contexts in [3]), or —as stated in HOSS ([9,10])— a set of links. Links are used to associate nodes, thus allowing the author to express semantic relationship. The system however is not ware of the semantics of the established links. Set of links create *documents*. Users (readers) are able to *browse* through the interconnected material by *traversing* links.

Taxonomic hypertext

Taxonomic reasoning is a particular kind of reasoning task that deals with the *comparison* and *classification* of highly similar nodes, in which an analyst viewing one node thinks not in terms of linking it to another node, but of including it in or excluding it from a set of related nodes [16]. The hierarchical structure is not a result of the data but is a way to structure the set of categories into which the artifacts are classified [17].

In general, the task of taxonomic reasoning has the following characteristics [16]:

- The information objects being manipulated are highly similar to one another along some dimensional attribute(s), so much so that the question of how to categorize or organize is not immediately obvious.
- Taxonoming reasoning develops descriptions of each item along a set of dimensions, but these dimensions are not fully defined when one begins the reasoning process.
- The basic activities that need to be supported to facilitate taxonomic reasoning are essentially set operations, such as sorting objects into sets based on their characteristics, looking together at the members of a single set, examining the different sets of which a single item is a member and generating new sets from old ones.

On the one hand, some nodes allow the analyst to select a perspective from which a given object is desired to be viewed. On the other hand, there are other nodes that require the analyst to sort the object into one of a set of disjoint categories. Actually, categories can be constructed in such a way that the structure remains a strict tree, and it is perspectives that cause the hierarchy to deviate from a strict tree, since they permit a given artifact to descend from several branches [17]. A taxonomic reasoning system contains three primary abstractions, the *specimen (*or *item),* the *taxon (*or *category)* and the *taxonomy*. A specimen has arbitrary content and attributes and represents a fact. A taxon, on the other hand, has no content but has arbitrary attributes. While a taxon can contain other taxons or specimens, a given taxon is contained in only one taxon. Finally, a taxonomy is a hierarchy of specimens and taxons.

Users do not traverse links, but rather *open* taxons, revealing their contents. Cycles are not allowed in taxonomic hypertext, since they do not have semantic meaning. Finding differences between two or more taxonomies (*delta calculation*) is a necessary aid for taxonomic workers.

Spatial hypertext

In the information analysis problem domain, the focus is on the process of structuring diverse material. Structuring and organization of information is a complicated intellectual process including activities such as collecting, comprehending and interpreting diverse types of information. During such activities, users tend to explore the structures they are creating and organize information by implicit and informal spatial methods formalizing *incrementally* their structures [4]. Taking advantage of the human perceptual system as well as spatial and geographic memory [5,6], these systems allow to express relationships by spatial proximity and visual cues. Such expression of relationships achieves differentiation of implicit from explicit structure hence permitting a smooth shift from informal to formal structure. Hypermedia systems providing such paradigm to information organization are referred to as spatial hypermedia systems.

Primary entities in spatial hypermedia systems include *objects*, *collections* and *composites* each usually appearing with a different visual symbol having different visual characteristics such as shape, colour, border, text, font, etc. Objects represent wrappers for arbitrary types of data. Object types are supported by means of explicit visual characteristics. Collections contain arbitrary amount of objects or collections. A set of objects or composites in a particular visual configuration form composites. Spatial hypermedia systems also provide visual and spatial means to denote relationships among entities. Consequently, these systems are able to recognize relationships by spatial arrangement, relationships by object type, relationships by collection, and relationships by composite. A special structure-finding process—the *spatial parser*—is responsible for discovering such relationships and suggesting the creation of new collections and composites to the user [13].

Hypertext Domain	**Abstraction**	**Behaviour**
Navigational	Node, link, anchor	Traversing link, opening node
Taxonomic	Taxon (category), item, taxonomy, perspective	Opening category, delta calculation
Spatial	Object, collection, composite	Spatial parsing

Table 1: Overwiew of abstractions and behaviors observed in the different various hypertext domains

Structural Computing

While hypertext domains *describe* a particular problem in information organization, hypertext systems attempt to provide the appropriate means to *support* information organization. Nevertheless, it has been shown that hypertext systems suitable for one domain exhibit insufficiencies for others. For example, attempting to model taxonomic hypertext with the use of a navigational hypertext system has been proven difficult and error prone. Spatial hypertext is difficult to discuss with the classic notions of hypertext such as "node" and "link". Domains such as spatial or taxonomic require conceptual foundations markedly different from those used to support navigational hypermedia manifesting, thus, a gap between hypermedia domain and system research. The incompetence of hypertext systems, led to a radical reconsideration of their conceptual foundations. Such re-examination and redesign of the foundations of hypertext systems is one central concern of structural computing [11,12].

Structural computing was originally inspired by research work on hypertext. A significant result in this area was the realization that there exist various *hypertext domains*, each one imposing different structural requirements, indicating different behavior and needing different services. In this context, hypertext recovers its original foundation as a knowledge organizing methodology; not only as a means for supporting navigation through inter-related documents.

Navigational, spatial, taxonomic and modeling are the most representative hypertext domains found in literature. This notion of hypertext domains had led to a layered architecture for hypertext systems, so called *component-based open hypertext systems* (*CB-OHS*). CB-OHSs are a realization of the structural computing approach. They provide an open set of components, called structure servers, each one providing the foundations to support the structural abstractions and behavior of a single hypertext domain, in a convenient and efficient way. Functionality that is common among all domains, is regarded part of the infrastructure (fig. 1).

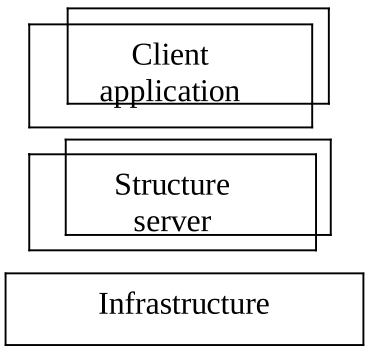


Fig. 1: Component-Based Open Hypertext Systems' Architecture

The *Callimachus* CB-OHS attempts to provide the framework in which the aforementioned hypertext domains co-exist and structure servers – providing new abstractions and services - for new domains can be developed. Given the complexity of development of structure servers for new hypertext domains, special attention has been given in the provision of suitable tools to facilitate such task.

One such tool is the *structure template*; a pattern of structure. The aim of structure templates is to maintain the specifications of the structural abstractions of different hypertext domains. Structure servers operate guided by these structure templates to provide domain specific functionality and constraints. Figure 2 outlines the conceptual architecture of *Callimachus* and the central role of structure templates. Each architectural entity has a specific role, which is briefly described below:

- Behavior. Behavior models the computational aspects of a domain, sub-domain or application. Behavior can be divided—depending on its purpose—into two main categories: *services*, which are available to clients through the use of a specific API and protocol (e.g. openNode, followLink, etc.), and *internal operations* that are used by the structure server internally for consistency reasons mainly (e.g. to affirm conditions and constraints, or to interpret abstractions in a suitable manner).
- Structure cache. Provides an immediate storage for the elements upon which behaviors operate.
- Infrastructure. This includes the fundamental hypermedia functionalities that are available to all other entities. Functionalities, such as naming, persistent store, and notification, constitute an essential part of the infrastructure.
- Template repository. A storage place for structure templates. Its main purpose is to hold structure definition as well as to support reusability and extensibility of structure among structure servers.
- Client. Any process that requests hypermedia functionality. Clients request hypermedia operations from one or more structure servers with the use of the appropriate APIs.

A structure template models the static aspects of the structural abstractions of a domain. The dynamic (or behavioral) aspects are incorporated into the *services* and *internal operations* modules.

Modeling static aspects (domain schema)

The methodology for defining the structure model of a domain consists of the specification and interrelation of *structural types*. A structural type is either an instantiation of a basic abstract class, the *Abstract Structural Element* (*ASE*), or a specialization of another existing structural type. The notion of the ASE is inspired by the *structure object*, proposed by Nürnberg [10] and is similar to the object (meta-)class in the object-oriented paradigm. The crucial characteristic of the ASE is that it is used to model a *relationship*. In Callimachus, *data* is conceived as a *degenerate form of relationship*: a relationship that is unable to relate.

Structure servers operate on *structural elements*, which are instances of structural types. The set of structural types defined by the designer at the same context, establish the template of the domain. During the definition of a new structure model, already existing templates may be reused or extended.

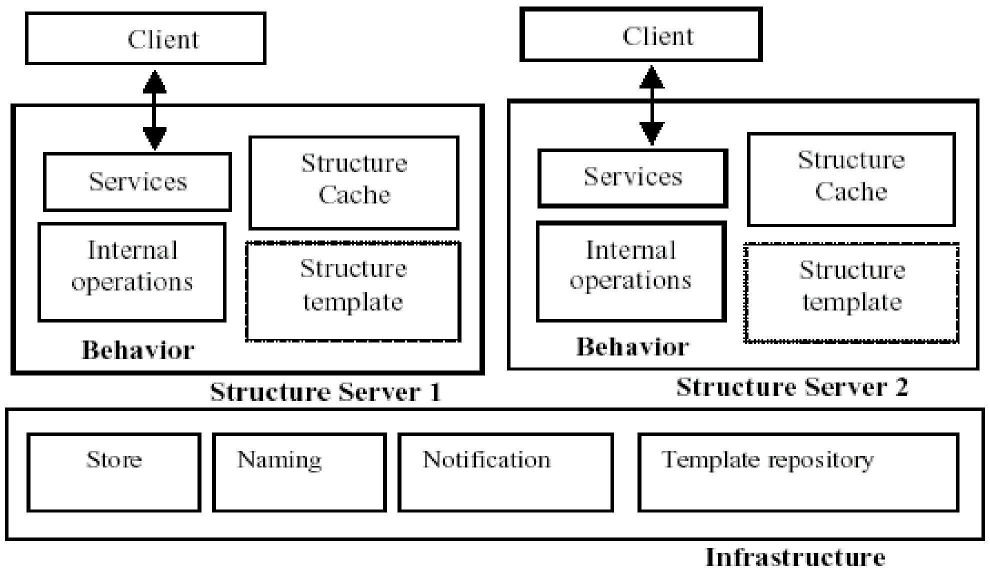


Fig. 2: Conceptual architecture of the Callimachus CB-OHS.

Structural types are identified by a system-defined unique *structural type identification* (*sid*) and a designer-specified unique name. A structure type may have an arbitrary number of *properties* and *end-*

sets. Structural elements are constrained by their structure type specification, regarding their endsets and properties. Each structural element is identified by a system-defined unique *object identification* (*oid*).

Properties are specified by a name, a data type and they may be single-valued or multi-valued. A special property, named *content*, may be defined in the context of a structural type. At the instantiation level, the value of this property is an address specification of an arbitrary amount of data. Since our purpose is to specify structure, not data, we do not define other semantics for that.

An endset of a structural element is a placeholder for oids of other structural elements. It is the most significant construct since it enables the grouping of related objects (fig. 3). An endset of a structure type has the following attributes:

- Name: unique among the endsets of the same structural type,
- Order number: an integer that designates the sequence of the endsets of a structural type,
- Total participation indication: a [min, max] notation that indicates the minimum and maximum number of oids that can participate in the endset.
- Multiplicity: a [min, max] notation that indicates the minimum and maximum number of occurrences the endset can participate in a structural element of that type.
- Configuration: whether or not the members of the endset bear a particular functional or structural relationship to one another (e.g. may form a set, list, queue etc).

Along an endset, a set of structural types is defined, called *s-set*, which configures the structure types whose instances may be end-points of the endset. An s-set may be *inclusive* or *exclusive*; an inclusive s-set specifies the allowed structure types, while an exclusive specifies the forbidden ones. A *partial participation indication* may be defined for each structure type in an inclusive s-set, indicating the minimum and maximum number of elements of this type that can participate in the endset. In addition, for each structural type in an s-set, a *cardinality indication* is specified defining the number of elements (of the structural type that is currently defined) to which a structural element of that type is allowed to be related (through that endset). This is similar to the cardinality ratio in the E-R diagrams that specifies the number of relationship instances that an entity instance can participate in, so a [min, max] notation is used as well.

As a last notation, an endset with more that one structural types in its s-set, may be characterized as *homogeneous* or *heterogeneous*. At the instantiation level, a homogeneous endset can include structural objects of only one type, while a heterogeneous endset may point to structural objects of different types simultaneously.

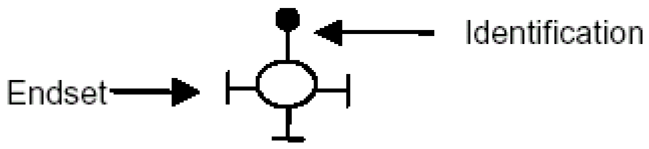


Fig. 3: A structural element with 3 endsets and identification

A *structure template* is the container of the structure type specifications that model a domain. A template is modeled itself as a structural object. Templates are stored in a special part of the infrastructure, called *template repository.* A template may be created from scratch or by utilizing existing templates in the repository. In this case, the new template is assigned a set of templates that inherits from.

Templates are represented in a formal specification language, since they include various attributes and complex relationships. A visual representation would be easier to use, but it is not easy to visually express all the details of the specifications. XML is used for such a task, as it enables composition and arbitrary details. Furthermore, it is a well-known language and can be manipulated easily.

Modeling behavioral aspects

Structure templates in Callimachus are not passive entities, but active constructs. They are *not controlled* by data, but *control data.* Templates *react* to changes in data, by *propagating* operations to all relevant abstractions, as denoted by the structure template. Different domains propagate operations in a different way. In particular, structural elements are considered contextualized and sensitive to events in their environment. They may *absorb* an event, meaning that they may initialize a specific computation or may propagate the same (or different) event to other, associated structural elements. A "routing table" for each structural type specifies how an event is broadcasted to the members of its endsets and to the structural elements to which it belongs. Such "micro-behavior" of structural elements (types) is used to achieve system wide (global) behaviors.

In Callimachus, the *propagation graph* captures how events generated by data are propagated within a structure. Figures 4 and 5 capture the propagation path for the delete operation in the navigational and taxonomic hypertext respectively. Arrows depict the scope of the propagation. Applying a delete operation on a node (fig 4, Node1), will result in deleting all associated anchors (fig. 4, Anchor1), but will not necessarily affect the link[1] (Link1) and surely not affect the destination nodes (fig 4. Node2). On the contrary, in taxonomic hypertext (fig. 5), a delete event will propagate recursively to all contained taxons and items resulting in their deletion. Different propagation schemas can be also witnessed when considering operations such as adding an attribute or copying a node or taxon.

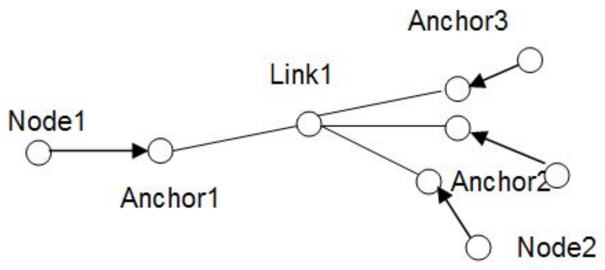


Fig. 4: Propagation graph of the delete operation in navigational hypertext.

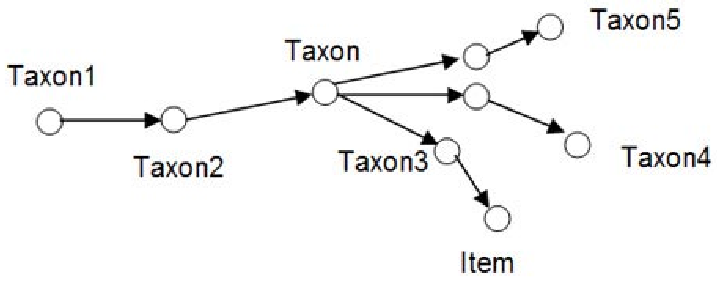


Fig. 5: Propagation graph of the delete operation in taxonomic hypertext.

Conclusions

The above discussion rises that structure templates and pattern types share some common characteristics. Both aim at modeling and handling relationships about data. Structure elements and patterns are treated as first class objects, which are explicitly separated from data items, even if they may act as "data" in different contexts. Although not discussed clearly in [18], we claim that patterns types, like structure templates, have also dynamic aspects. For example, rules denote transitions from a "left-

[1] If anchors of other nodes act as source of the same link, then the link should not be deleted.

hand" to a "right-hand" side, clusters and decision trees require certain traversal and manipulation operations, serial and parallel episodes drive different occurrences of events in time.

The above observations conceal a number of common requirements that should be laid in order for a management system for patterns to be developed. We perceive three such requirements: Arbitrary granularity, valence specification and operation propagation.

High level units may be constructed from or divided in lower level ones in a recursive manner. The "minimal abstraction" provided by the system should be carefully decided in order to support arbitrary granularity in modeling data item relationships. The same unit may be treated either as "individual" or "composite" in different contexts.

The term valence is borrowed from chemistry. We use it to denote "what" can be related according to a given pattern or structure type. The definition language of the management system should support the declaration of constraints and validation rules, stemming from an application domain, so that only valid instances could be created.

Besides the static aspects of a pattern or structure type, its behavior may be modeled as well. O p-eration propagation seems to be a promising approach in this direction. Units receive external "stimulus" (from other units, or the context) and react by either "absorbing" it internally, propagating it to other units, or both. "Routing tables" for each type may be defined in order to incorporate the necessary behavioral knowledge of the application domain.

Another interesting issue is the interoperability of patterns inferred from different application domains. For example, a pattern from the mathematics domain maybe tested for validity against a music database. In structural computing, the same problem is approached by the definition of transformation rules that convert a structural element of one hypertext domain to an element for a different domain.

In conclusion structural computing and PBMS, although originating from different research areas, both are looking for answers in similar kinds of problems. The common objective to emerge the significance of the relationships on data, rather than the data items themselves, may trigger the interchange of ideas, methodologies and practices between the two communities.

References

1. Bush, V.: As We May Think. Atlantic Monthly, pages 101–108, July, 1945.
2. Halasz F, Schwartz M. The Dexter hypertext reference model, Commununications of the ACM 37 (2) (1994) 30–39.
3. Leggett J. J. and Schnase, J. L. Viewing Dexter with Open Eyes. Communications of the ACM, 37(2):76–86, 1994.
4. Marshall CC, Rogers RA, Two years before the mist: experiences with Aquanet, ECHT '92 Proceedings, 1992, p. 53–62.
5. Marshall CC, Shipman III FM. Coombs JH. VIKI: spatial hypertext supporting emergent structure, ECHT '94 Proceedings, 1994, p. 13–23.
6. Marshall CC. Shipman III FM, Spatial hypertext: designing for change, Communications of the ACM 38 (8) (1995) 88–97.
7. Nelson, T.: A File Structure for The Complex, The Changing and The Indeterminate. In Association for Computing Machinery Proceedings of the 20th National Conference, pages 84–100, 1965.
8. Nürnberg P, Leggett J, Schneider E. As we should have thought, Hypertext '97 Proceedings, Southampton, UK, ACM, 1997, p. 96–101.
9. Nürnberg P, Leggett J. A vision for open hypermedia systems, Journal of Digital Information 1 (2) (1999) 207–248.
10. Nürnberg, P. J., "HOSS: An Environment to Support Structural Computing", Ph.D. dissertation, Dept. of Computer Science, Texas A&M University, College Station, TX, 1997.
11. Reich, S., Anderson, K.: Open Hypertext Systems and Structural Computing, 6th International Workshop, OHS-6, 2nd International Workshop, Sc-2, San Antonio, Texas, USA, May 30-June 3, 2000, Proceedings. Springer 2000
12. Reich, S., Tzagarakis, M. and De Bra, P.: Hypermedia: Openness, Structural Awareness, and Adaptivity, International Workshops OHS-7, SC-3, and AH-3, Aarhus, Denmark, August 14-18, 2001. Revised Papers. Springer 2002
13. Reinert, O., Bucka-Lassen, D., Pedersen, C.A., Nurnberg, P.J. "CAOS: A Collaborative and Open Spatial Structure Service Component with Incremental Spatial Parsing", Proceedings of the ACM 1999 Conference on Hypertext, 1999, pp. 49-50.
14. Trigg, R., Suchman, L. and Halasz, F.: Supporting Collaboration in Note-Cards. In CSCW 86 Proceedings, pages 153–162, Austin, Texas, USA,1986, ACM.
15. Tzagarakis, M., Avramidis, D., Kyriakopoulou, M., Schraefel, M. M. C., Vaitis, M., Christodoulakis, D.: "Structuring Primitives in the Callimachus Component-based Open Hypermedia System", Journal of Network and Computer Applications, Academic Press, to appear.

16. Van Dyke Parunak H. Don't link me in: set based hypermedia for taxonomic reasoning, Hypertext '91 Proceedings, San Antonio, TX, USA, ACM, 1991, p. 233–42.
17. Van Dyke Parunak H. Hypercubes grow on trees (and other observations from the land of hypersets), Hypertext '93 Proceedings, Seattle, WA, USA, ACM, 1993, p. 73–81.
18. Vazirgiannis, M. et al.: "A Survey on Pattern Application Domains and Pattern Management Approaches", PANTA (IST-FET project) Technical Report Series, PANDA-TR-2003-01, February 2003.

What's new in Querying, Query Processing and Optimization in PBMS?

Ilaria Bartolini[1], Paolo Ciaccia[1], Marco Patella[1], and Yannis Theodoridis[2]

[1] DEIS – University of Bologna, Italy
e-mail: {ibartolini, pciaccia, mpatella}@deis.unibo.it

[2] CTI and University of Piraeus, Greece
e-mail: ytheod@cti.gr

Abstract: Representing, storing, and manipulating useful patterns extracted from raw data is an emerging field of research in data management The architecture of a so-called Pattern-Base Management System (PBMS) should definitely include modules for pattern definition/manipulation languages, appropriate indexing mechanisms and efficient query processing, evaluation and optimization techniques. In this paper, we provide a preliminary overview of the above issues and discuss efficient solutions.

1. Introduction

In this paper, we consider the major issues related to querying and query processing in a, so called, Pattern-Base Management System (PBMS), that handles patterns extracted from large databases.

The outline of the paper is as follows: In Section 2, we briefly review what has emerged so far concerning the logical modeling of patterns and then, in Section 3, we classify operation types depending on the type of patterns they apply to and on the type of relationships between patterns and between patterns and raw data they exploit. In Section 4, issues concerning the declarative languages that supported by the PBMS, are reported. A set of preliminary examples of pattern types, pattern instances and queries is also presented in that section. Section 0 introduces some query optimization issues based on query examples presented earlier, in Section 4. Finally, Section 6 highlights the relevance of some operation types which, according to our view, are peculiar to PBMSs and as such deserve careful consideration, and summarizes some open issues for future work.

2. Basics from a Logical Model for Patterns

In [6], a logical model for patterns is proposed and the concept of Pattern-Base Management Systems (PBMS) is introduced. Restricting our attention to those aspects which can have a relevant impact on the physical level, we list the following:

1. The PBMS will manage a variety of *pattern types*, which cannot be a-priori defined; the extension of each pattern type will consist of a collection of *patterns* sharing similar characteristics (i.e. they fit the pattern type definition).
2. A pattern type *pt* is a quintuple (*n*, *ss*, *ds*, *ms*, *f*), where *n* is the name of the pattern type; *ss*, *ds*, and *ms* (called, respectively, structure schema, data source schema, and measure schema) are types in *T* (the set *T* of types includes all the base types together with all the types recursively defined by applying a type constructor to one or more other types); *f* is an formula, written in a given language, which refers to type names appearing in the source and in the pattern schemas.
3. A pattern *p* instance of a pattern type PT is a quintuple (*pid*, *s*, *d*, *m*, *e*), where *pid* (pattern identifier) is a unique identifier for *p*; *s* (structure) is a value for type *ss*; *d* (data source) is a dataset whose type conforms to type *ds*; *m* (measure) is a value for type *ms*; *e* is obtained by the formula *f* by instantiating each type name appearing in *ss* with the corresponding value specified in *s*.

Besides managing patterns, a PBMS should also take a set of *pattern (semantic) relationship types* (e.g. aggregation, refinement, generalization) into account, which will be used to model a pattern base according to the specific application needs.

It is also convenient to take into account the fact that some interesting relationships between patterns arise because of the presence of *data source relationships* that hold between the corresponding data source schemata *ds*. For instance, two sets of association rules might be related because they have been mined from two data sets corresponding to different stores of a same chain.

3. Classification of Pattern Operations

For query processing purposes, patterns are classified according to how they are constructed and what they are used for (Fig. 1).

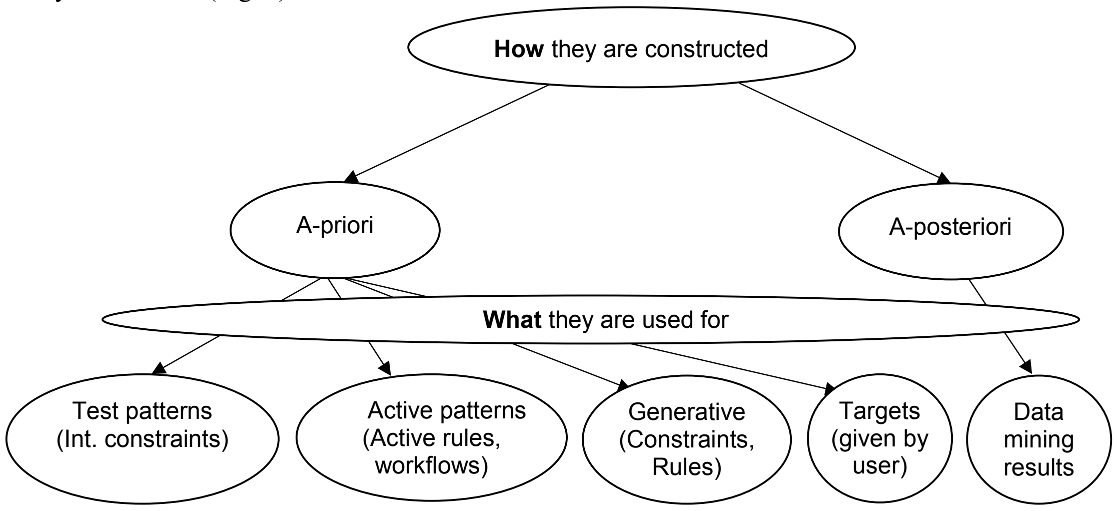

How they are constructed
A-priori
A-posteriori
What they are used for
Test patterns (Int. constraints)
Active patterns (Active rules, workflows)
Generative (Constraints, Rules)
Targets (given by user)
Data mining results

Fig. 1. Pattern classification schema

- *A-posteriori patterns*: patterns created in the PBMS by means of a processing on the raw data, typically as results of a data mining task.
- *A-priori patterns*: patterns existing a priori in the raw dataset (e.g. integrity constraints), being enforced by DBMS administrator or user.

From the query processing point of view, a-posteriori patterns typically require some kind of processing to be performed on the raw data in order to extract their structure and measure components, whereas a-priori patterns are usually compared against raw data to produce a score assessing how well a pattern is representative of (a part of the) raw data.

We now analyze the main patterns relationship types we are interested in for query processing issues (see also [6]).

- *Composition*: A complex pattern can be obtained by assembling other patterns; more precisely, an aggregated pattern has a structure schema which is defined in terms of one or more pattern types.
- *Refinement*: A refinement is a relationship existing between patterns at different abstraction or granularity levels.
- *Specialization*: A pattern type pt1 specializes another pattern type pt2 when the structure schema, the source schema, and the measure schema of pt1 specialize the structure schema, the source schema, and the measure schema, respectively, of pt2.
- *Induced relationships*: Relationships existing between subsets of the data space can induce relationships on patterns related to such data. Relationships of this type can also lead to refinement relationships. As an example, consider a functional dependency between two features product and category in the data space: product → category. Such relationship would naturally induce a number

of relationships between patterns generated over the data, e.g. between rules built on categories and rules built on products. This relationship can also be viewed as a refinement.

Such relationships between patterns, along with the obvious relationship between patterns and the data they are extracted from (for a-posteriori patterns) or against which they have to be matched (for a-priori patterns), define a classification of the different operation types a PBMS has to deal with. In the following we will present examples of such classification.

3.1 Basic operations

A basic operation between patterns is that of **comparison**: Two patterns of the same (simple) type can be compared to compute a score s, $s \in [0,1]$, assessing their mutual similarity. The similarity between two simple patterns is computed as a function of the similarity between both the structure and the measure components:

pattern_similarity = f(structure_similarity, measure_similarity)

Supposed two patterns have the same structure, the measure of similarity naturally corresponds to a comparison of the patterns' measures, e.g. by aggregating differences between each measure [2]. In the general case, however, the patterns to be compared have different structures, thus a preliminary step is needed to *reconcile* the two structures to make them comparable.

3.2 Operations on PB-DB relationships

These operations allow crossing the boundary between the PBMS and the DBMS, relating patterns to data from which they are generated or with which they have to be matched.

– *PB-DB relationships for a-posteriori patterns*. This class contains all the operations which are used to create patterns from the raw data. Typical examples include data mining algorithms as extraction of association rules for market-basket analysis, clustering, etc. Such operations are typically used to populate the Pattern Layer of the PBMS.
– *PB-DB relationships for a-priori patterns*. As above anticipated, a-priori patterns are patterns which already exist in the PBMS and that are compared (e.g. matched) to the raw data at hand. Basic operations belonging to this class include:
  a. Matching a pattern against a subset of the data space (*matching*), obtaining the data objects along with their score expressing how well they fit the given pattern.
  b. Matching a data object against a set of patterns of a given type (*classification*), obtaining a score for each pattern, expressing how well each pattern is suited to represent the given datum.
  c. Obtaining the measures for a given pattern with respect to a set of data (*measuring*); given the pattern structure, the user could ask to compute its measures with respect to a subset of the raw data. As an example of this class of operations, just consider the computation of support and confidence measures for an association rule over a set of data.

3.3 Operations involving composition

In a complex pattern, the structural component includes the schemes of simple component patterns, whereas the measure part may include values defined as expressions of the component pattern schemes [6]; an obvious generalization allows to aggregate complex patterns, thus defining a multi-level pattern hierarchy. A basic operation belonging to this class is the comparison of complex patterns. The similarity score s between two complex patterns of the same type is computed starting from the similarity between component patterns, then the scores obtained for each sub-pattern are aggregated, using an *aggregation logic*, to determine the overall similarity of the two patterns [2]. The aggregation logic may be very simple, just an expression combining numerical values, or a more complex one, if constraints and/or transformation costs are to be considered: For example, a suitable "matching" between components patterns might be needed. The aggregation logic can also involve a variety of transformations, each with an associated cost, and the overall similarity score is obtained as the maximum score obtained by applying all the possible transformations to the component patterns.

3.4 Operations involving refinement of patterns

Here, the most common operations are zooming in/out on different granularity/abstraction levels. For a-priori patterns, these correspond to multi-resolution queries: Processing of such queries may require access to raw data, depending on the relationships existing between patterns. As an example, consider a hierarchical clustering algorithm yields a *dendrogram*, where the nested grouping of clusters is represented [3]. At each granularity level, different clusters can be determined, see Fig. 2. Measures for clusters at higher abstraction levels (i.e. with lower granularity) can be obtained with simple computations from measures for clusters at lower levels, thus without need to access the raw data.

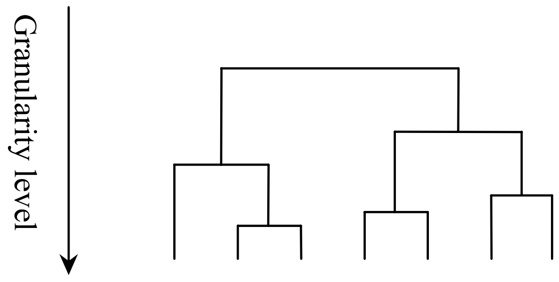


Fig. 2. A dendrogram

3.5 Operations involving specializations (or hierarchies)

Operations in this class allow the user to navigate the types hierarchy by generalizing (i.e. moving up the types hierarchy) or specializing (i.e. moving down the types hierarchy) a pattern. These are simple operations and we will not detail them further.

3.6 Operations on induced relationships

Operations belonging to this class act on patterns which have no direct relationship, but whose relationship is induced by relationships existing on the data such patterns are associated to. This may require to access the raw data in order to compute measures for patterns. For example, consider the relationship between association rules on categories and rules on products: if we want to compute the exact confidence of a rule on a given category, we cannot devise a formula to obtain it from measures of rules built on products of that category, but the original data have to be accessed.

4. Querying a Pattern Base

A DBMS (PBMS) is basically a system that stores and retrieves data (respectively, patterns) upon user's requests. The two fundamental features that are supported by a DBMS are the data model and the high-level languages for defining, manipulating and retrieving data. We are considering that the PBMS must provide the same set of declarative languages for defining, manipulating and retrieving pattern types and patterns.

4.1 Pattern Definition Language (PDL)

PDL is placed in the pattern type layer according to the PBMS architecture proposed in [6]. This language enables PBMS users to manipulate pattern types that are stored in the PBMS catalog. PDL must implement a set of constraints and checks that are related to the underlying patterns. For example, if a user posts a pattern type deletion the PBMS must check for existing pattern instances and abort the operation if associated patterns are found. The definition of PDL is an open issue.

4.2 Pattern Manipulation Language (PML)

PML should be able to support insertions, deletions and updates for patterns. The usage of PML in manipulating a-priori patterns is essential. Users should be given the tools to insert an a-priori pattern through a declarative way and PBMS should perform all the semantic checks before storing the pattern in the pattern base. For example, the condensed expression of a pattern type has its own structure and every value that is assigned to it must conform to that structure. Thus, the syntax and the semantic checks must be performed in multiple levels before a pattern acquires the grant for storing.

4.3 Pattern Retrieval (or Query) Language (PRL or PQL)

Queries in PBMS are divided in two main categories: queries that apply to pattern types and queries that apply to pattern instances. The query processor mostly derives pattern type queries at the semantic analysis phase, that is, after a query has been syntactically analyzed it is further checked for inconsistencies between the entities that it contains and the entities that exist in the PBMS catalog. Next, we give some examples for queries of that type, partially taken from [6].

Name	Structure Schema	Data source Schema	Measure Schema	Function
Association Rule	RECORD(head: SET(STRING), body: SET(STRING))	BAG(transaction: SET(STRING))	RECORD(confidence: REAL, support: REAL)	head $\cup$ body $\subseteq$ transaction
Interpolating Line	SET(sample: RECORD(x: REAL, y: REAL))	RECORD(a: REAL, b: REAL)	fitting: [0..1]	$y = a \cdot x + b$
TimeSeries	RECORD(curve: LIST(y: REAL), position: TIMESTAMP, shift: REAL, gain: REAL)	LIST(sample: REAL)	similarity: [0..1]	sample[position + i − 1] = shift + gain $\times$ curve[i], $\forall i$: $1 \leq i \leq$ *length*(curve)

Table 1. Examples of pattern types

Pattern Identifier (pattern type)	Structure	Data source	Measure	Expression
512 (Association Rule)	(head = {'Boots'}, body = {'Socks', 'Hat'})	'SELECT SETOF(article) FROM sales_in_US GROUP BY transactionId'	(confidence = 0.75, support = 0.15)	{'Boots', 'Socks', 'Hat'} $\subseteq$ transaction
513 (Association Rule)	(head = {'Hat'}, body = {'Boots'})	'SELECT SETOF(article) FROM sales_in_Canada GROUP BY transactionId'	(confidence = 0.60, support = 0.35)	{'Boots', 'Hat'} $\subseteq$ transaction
456 (TimeSeries)	(curve = (y = 0, y = 0.8, y = 1, y = 0.8, y = 0), position = 12, shift = 2.0, gain = 1.5)	'colorado.txt'	similarity = 0.83	sample[12] = 2.0, sample[13] = 3.2, sample[14] = 3.5, sample[15] = 3.2, sample[16] = 2.0

Table 2. Examples of patterns

Apart from the query processor that submits queries to the system catalog during the semantic checking phase, users may also query the system catalog of the PBMS. The PBMS can be seen as a repository for pattern types and patterns, thus queries submitted to the system catalog may be of interest for a group of users. Examples of system catalog queries posted by the query processor will be presented along with queries on patterns. We use a table to demonstrate the result of every query but this does not imply that the query result will be a set of tuples as in the relational model.

4.3.1 Queries on pattern types

– *Q1: Retrieve all pattern types of the PBMS:* Q1 will return the entire system catalog. The type of this query result is unknown. Will it be a set of objects of type "pattern type" or a set of pattern type names that will be materialized upon user's or application's request? The result, according to the first option, would be identical to Table 1.
– *Q2: Retrieve the measures of pattern types:* Q2 shall return the measure schema of every pattern as shown in Fig. 3 (a).
– *Q3: Retrieve names and structure schemata of pattern types extracted by datasets of type bag (i.e. the Data source schema is like BAG%):* This query has a selection predicate on the value of the Data source schema component. The query language should have the ability to express such queries that refer to the internal structure and to the values of the pattern type components. The query result of this query is depicted in Fig. 3 (b).

Measure Schema
RECORD(confidence: REAL, support: REAL)
fitting: [0..1]
similarity: [0..1]

(a) result of Q2

Name	Data source Schema
Association Rule	BAG(transaction: SET(STRING))

(b) result of Q3

Fig. 3. Results of queries on pattern types

4.3.2 Queries on patterns

– *Q4: Retrieve patterns of type "Association Rule":* This query returns patterns that are instances of the "Association Rule" type and can be determined in two alternative ways. We will use the notation of SQL to illustrate the example.

```
select * from "Association Rule";
```

The above query implies a relational flavor, that is, every pattern type is a first-class object in the PBMS.

```
select * from patterns
where pattern_type = "Association Rule";
```

On the other hand, this alternative query implies that all pattern instances are members of the unique and general entity "Pattern". Thus, the part "select * from Pattern" is always implied in every query.
– *Q5: Retrieve the name of types and the value of all Measures named "confidence" where value of confidence must be greater than 0.70:* The query result is shown in Fig. 4 (a).
– *Q6: Retrieve the head of patterns with type "Association Rule" (AR1) and the body of patterns with type "Association Rule" (AR2) having the body of the first to be equal with the head of the second:* The expected query result is shown in Fig. 4 (b).

Pattern Identifier (pattern type)	**Measure**
512 (Association Rule)	(confidence = 0.75, support = 0.55)

(a) result of Q5

AR1.head	**AR2.body**
{'Hat'}	{'Socks', 'Hat'}

(b) result of Q6

Fig. 4. Results of queries on patterns

5. Query optimization

In this section we present some preliminary issues concerning the query evaluation process in the PBMS. Again, the use of terms like "scan", "select" and "project" are taken from the relational lingua franca and do not impose any restriction on future definitions of the PBMS data model operations. We will use example Q5, taken from the previous section, to demonstrate some preliminary query evaluation plans. For example, Fig. 5 (a) and (b) illustrate a primitive and an improved, respectively, execution plan for Q5.

step 1. Scan the Pattern Base to find patterns that satisfy the condition "Measure.confidence > 0.70".

step 2. Project the value of measure from the patterns returned in Step 2.

step 3. Construct the query result

(a) naïve execution plan for Q5

step 1. Scan system catalog to find pattern types having "confidence" in their measure schema.

step 2. In the retrieved pattern types from Step 1, identify the type of measure "confidence" and check if the comparison with the const value "0.70" is valid. This implies static type checking.

step 3. Assuming an index on patterns.PTName, perform index scan in the pattern space and retrieve only those patterns that are instances of pattern type(s) of Step 1 and satisfy the condition "Measure.confidence > 0.70".

step 4. Project the value of measure from the patterns returned in Step 3.

step 5. Construct the query result.

(b) improved execution plan for Q5

Fig. 5. Results of queries on pattern types

6. Final Remarks and Summary

In this paper, we have presented some preliminary aspects on querying and query processing in PBMS. In the following, we address some issues that should be addressed in future work.

Pattern comparison: As anticipated in Section 3.1, comparison between patterns requires matching both the structural and the measure components of patterns [2].

Comparison of complex patterns: The comparison of complex patterns (see Section 3.3) entails the use of aggregation logic. Efficient evaluation of aggregation algorithms may require the indexing of component patterns, for example a metric index like the M-tree [1] can be used if the (dis-)similarity between component patterns is a metric. Other issues arise if one considers complex matches between component patterns, e.g. *M*-to-*N* matching, matching based only on a subset of component patterns, etc. Moreover, one should also consider that a number of transformations can be applied in order to obtain a better matching. Such transformations can considerably alter the complexity of matching. In this complex scenario, the use of approximate strategies could significantly reduce evaluation costs.

Obtaining measures for finer/coarser patterns: These operations allow the user to zoom in/out on different granularity/abstraction levels. Measures for coarser patterns can be obtained by appropriately combining the measure components of finer patterns. In order to reduce accesses to the DBMS, approximate results and additional knowledge on the object domain can be used. Similar arguments also apply to the case where measures are requested for finer patterns, since they can be estimated starting from measures for coarser patterns.

Other issues include:

- Operators similar to "Project", "Select", "Join", etc. should be defined for the structure and values of pattern types and patterns.
- Type checking and casting mechanism for PBMS need to be defined, as well as the level of strictness that it will provide.
- Definition of the intermediate query results types should be addressed.
- Additional metadata for pattern types and patterns could be considered.

References

[1] P. Ciaccia, M. Patella, and P. Zezula. M-tree: An efficient access method for similarity search in metric spaces. *Proceedings of the 23rd VLDB International Conference*, pp. 426-435, 1997

[2] V. Ganti, J. Gehrke, R. Ramakrishnan, Wei-Yin Loh. A Framework for Measuring Changes in Data Characteristics. *Proceedings ACM Symposium on Principles of Database Systems*, Philadelphia, PA, pp. 126-137, 1999.

[3] A. K. Jain, M. N. Murty, and P. J. Flynn. Data clustering: a review. *ACM Computing Surveys*, 31 (3), pp. 264-323, 1999.

[4] E. Keogh. Exact Indexing of Dynamic Time Warping. *Proceedings of the 28th VLDB International Conference*, pp. 406-417, 2002.

[5] E. Keogh, M. J. Pazzani: Relevance Feedback Retrieval of Time Series Data. *Proceedings of ACM SIGIR Conference*, pp. 183-190, 1999.

[6] S. Rizzi et al. Towards a Logical Model for Patterns. *Under submission*, 2003.

Correct Algorithms for the Comparison of Complex Patterns

Ilaria Bartolini, Paolo Ciaccia, and Marco Patella

DEIS - IEIIT-BO/CNR, University of Bologna, Italy
{ibartolini,pciaccia,mpatella}@deis.unibo.it

Abstract. The comparison of complex patterns, i.e. patterns that are obtained by recursively aggregating other patterns, poses serious challenges because of the different ways the component patterns can be matched. In this paper, we investigate the problem of comparing set of patterns when constraints are imposed on the matching between component patterns. This is motivated by a real world example, namely the retrieval of images in a region-based image retrieval system, where images represent complex patterns that are composed of regions, i.e. base patterns. We present both sequential and index-based exact algorithms for solving the problem, and experimentally evaluate them on a medium-size data set.

1 Introduction

The massive quantity of data produced every day by both industrial and scientific applications poses new challenging requirements to DBMS systems. Such huge amount of data is unlikely to be useful for end users, and automated processing techniques (such as data mining, pattern recognition, and knowledge extraction techniques) are needed in order to reduce such raw data to a compact, manageable, set of knowledge artifacts (e.g. clusters, association rules, time series). A compact and rich in semantics representation of raw data is called a *pattern* [13, 4]. The problem of storing and querying patterns in an effective and efficient way is the focus of so-called *Pattern-Base Management Systems* (PBMSs). Among the several issues that a PBMS has to address (modeling, storage, and retrieval of patterns), we believe that one of the most important operations that should be supported is that of comparison. The comparison between two patterns entails the computation of a score s, $s \in [0, 1]$, assessing their mutual similarity. Given the definition of similarity between patterns, the user may be interested in finding the patterns which are most similar to a given (query) one. More precisely, given a query pattern q and an integer value k, a *best matches query* returns the k patterns having the highest similarity score with respect to q, according to the similarity measurement implemented.

The comparison between *complex* patterns, i.e. patterns obtained by assembling other patterns to obtain a *part-of* hierarchy (for example, a *clustering* pattern is obtained as the composition of *cluster* patterns) is particularly challenging. The similarity score s between two complex patterns is computed starting from the similarity between component patterns, then scores obtained for each sub-pattern are aggregated, using an aggregation logic, to determine the overall similarity of the two patterns [9]. The aggregation logic may be very simple, just an expression combining numerical values, or a more complex one, if constraints and/or transformation costs are to be considered: For example, a suitable "matching" between components patterns might be needed. The aggregation logic can also involve a variety of transformations, each with an associated cost, and the overall similarity score is obtained as the maximum score obtained by applying all the possible transformations to the component patterns.

In this paper, we provide correct sequential and index-based algorithms to solve best matches queries for complex patterns consisting of a set of base patterns when constraints on the matching between component patterns exist. In Section 2 we introduce a (simplified) framework for modeling patterns. Then, Section 3 provides a motivating example, drawn from the world of image retrieval, for the problem of best matches queries. The problem is precisely formalized in Section 4 and both sequential and index-based correct algorithms for its solution are provided in Section 5. Section 6 shows some preliminary results obtained over a real data set and Section 7 concludes the paper.

2 A Model for Patterns

The following framework for modeling patterns is a simplified version of the model proposed in [13, 4], that we adapted to our needs.

A *pattern type pt* specifies the intensional form of patterns, and is represented by a quintuple $pt = (n, ss, ds, ms, f)$, where:

- n is the name of the pattern type.
- The *structure schema ss* describes the structure of the pattern instances of pt, thus defining the space in which patterns can be defined.
- The *source schema ds* defines the schema of the data set from which instances of pt are constructed.
- The *measure schema ms* describes the measures that quantify the quality of the representation of source data achieved by the pattern.
- The *formula f* describes the relationship between source space and pattern space and, thus, carries the semantics of the pattern.

Example 1. The pattern type "2-D cluster" can be defined as follows:

$$\begin{aligned} n &: \mathsf{2DCluster} \\ ss &: \mathsf{RECORD}(radius\text{: }\mathsf{REAL},\ center\text{: }\mathsf{RECORD}(cx\text{: }\mathsf{REAL},\ cy\text{: }\mathsf{REAL})) \\ ds &: \mathsf{SET}(point\text{: }\mathsf{RECORD}(x\text{: }\mathsf{REAL},\ y\text{: }\mathsf{REAL})) \\ ms &: numberOfPoints\text{: }\mathsf{INTEGER} \\ f &: (x - cx)^2 + (y - cy)^2 \leq radius^2 \end{aligned}$$

□

A pattern p, instance of a pattern type pt, is defined as a quintuple $p = (pid, s, d, m, e)$, where:

- pid (pattern identifier) is an unique identifier for p.
- The *structure s* is a value of the structure schema ss.
- The *data set d* conforms to type ds.
- The *measure m* is a value of type ms.
- The *expression e* is obtained by opportunely instantiating the formula f.

Example 2. A cluster of type 2DCluster (see Example 1) can be represented as follows:

$$\begin{aligned} pid &: \mathsf{324} \\ s &: \mathsf{RECORD}(radius\text{: }2.3,\ center\text{: }\mathsf{RECORD}(cx\text{: }35,\ cy\text{: }1570)) \\ d &: \text{'SELECT E.age, E.salary} \\ &\quad\ \text{FROM Employee as E'} \\ m &: numberOfPoints\text{: }184 \\ e &: (\mathsf{E.age}-35)^2 + (\mathsf{E.salary}-1570)^2 \leq 2.3^2 \end{aligned}$$

□

Note that composition of patterns can be obtained by inserting pattern types in the structure schema of a pattern type pt.

Example 3. A clustering is just a set of clusters (see Example 1) having a measure expressing the *validity* of the obtained clusters:

$$\begin{aligned} n &: \mathsf{2DClustering} \\ ss &: clusters\text{: }\mathsf{SET(2DCluster)} \\ ds &: \mathsf{SET}(point\text{: }\mathsf{RECORD}(x\text{: }\mathsf{REAL},\ y\text{: }\mathsf{REAL})) \\ ms &: validity\text{: }\mathsf{INTEGER} \\ f &: \end{aligned}$$

□

The similarity between two simple patterns of the same pattern type is computed as a function of the similarity between both the structure and the measure components:

$$sim(p_1, p_2) = f(sim_{struct}(p_1.s, p_2.s), sim_{meas}(p_1.m, p_2.m))$$

where with $p.s$ and $p.m$ we indicate the structure and the measure for pattern p, respectively. If the two patterns have the same structural component, then $sim_{struct}(p_1.s, p_2.s) = 1$, and the measure of similarity naturally corresponds to a comparison of the patterns' measures, e.g. by aggregating differences between each measure [9]. In the general case, however, the patterns to be compared have different structural components, thus a preliminary step is needed to reconcile the two structures to make them comparable. Computing the similarity between complex patterns, i.e. instances of a complex pattern type, in the general case is a two step process:

Matching: Component patterns of a pattern are associated to component patterns of the reference (query) pattern, i.e. by only considering "best" couplings (matches).

Combining: The overall similarity between the two patterns is computed by combining similarity scores corresponding to matched component patterns.

3 Motivating Example

The goal of content-based image retrieval (CBIR) systems is to define a set of properties (*features*) able to effectively characterize the content of images and then to use such features during retrieval in order to provide effective and efficient access to image databases based on content. To increase the effectiveness of image retrieval, in recent times a number of *region-based* image retrieval systems has been presented [5, 12, 1, 14], which "fragment" each image into regions, i.e. sets of pixels sharing common visual characteristics, like color and texture. Similarity between images is then assessed by computing similarity between pairs of regions and combining the results at the image level.

Conceptually, each image is represented as a set of component regions. By considering the model of Section 2, we can represent each region as a simple pattern and the overall image as a set of region patterns. This way, the problem of finding the images that most resemble a given query one can be modeled as a best matches query over the space of image patterns. In particular, the process of similarity assessment between images perfectly fits the matching/combining paradigm introduced in Section 2 (see Figure 1).

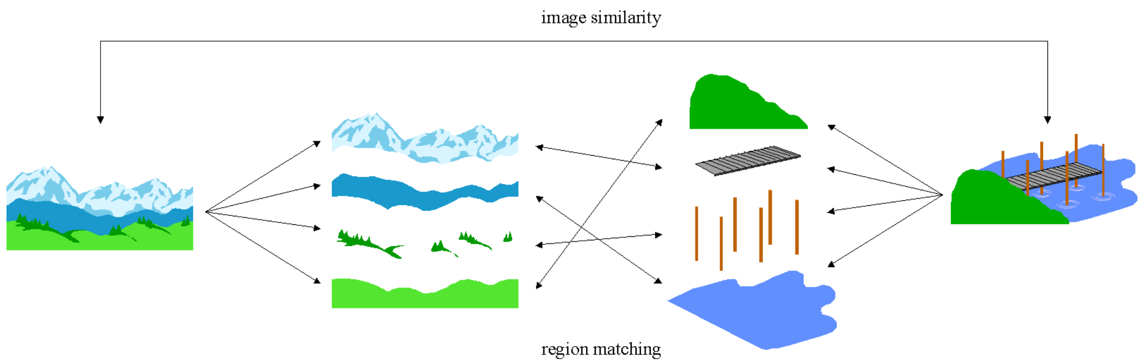


Fig. 1. In region-based systems, similarity between images is assessed by taking into account similarity between matched regions.

Region matching algorithms have only recently emerged as a need for region-based CBIR systems. Existing systems [5, 12, 14], however, use naïve heuristic matching algorithms when associating regions of the images being compared, thus obtaining incorrect results.[1]

The criterion used to assess the similarity between two regions vary from system to system. For example, the WINDSURF system [1] segments images into sets of pixels that are homogeneous for color and texture by using the Discrete Wavelet Transform (DWT, [7]) and a fuzzy c-means algorithm. Each region is then represented as an elliptical cluster in the HSV space and the similarity between regions is computed by taking into account both differences in the color and textures descriptors (the pattern structure) by way of the Bhattacharyya distance and in their relative size (the pattern measure). For more details, see [1].

4 The Problem of Optimal Matching

Given a reference (query) complex pattern cp_q, composed of a set of patterns $\{p_{q_1}, \ldots, p_{q_n}\}$, and a pattern p_s, also composed of a set of patterns $\{p_{s_1}, \ldots, p_{s_m}\}$, the problem of *optimal matching* consists in associating (matching) each pattern p_{q_i} of cp_q to a pattern $p_{s_j} = \Gamma_s(p_{q_i})$ of cp_s (possibly, no pattern is associated to p_{q_i}, i.e. $\Gamma_s(p_{q_i}) = \emptyset$) such that the overall similarity score between patterns cp_q and cp_s, $sim(cp_q, cp_s)$, is maximized. Similarity between base patterns is assessed by way of the $sim_{base}(p_{q_i}, p_{s_j})$ function. Every $\Gamma_s()$ has to satisfy the following constraint: Two patterns of cp_q cannot be associated to the same pattern of cp_s, therefore if $p_{q_i} \neq p_{q_j}$ and $\Gamma(p_{q_i}) = \Gamma(p_{q_j})$, it is $\Gamma(p_{q_i}) = \Gamma(p_{q_j}) = \emptyset$. Similarity between complex patterns is computed by taking into account similarity between associated base patterns using a monotonic function PM_{sim}, i.e. $sim(cp_q, cp_s) = PM_{sim}(sim_{base}(p_{q_1}, \Gamma_s(p_{q_1})), \ldots, sim_{base}(p_{q_n}, \Gamma_s(p_{q_n})))$. The only requirement for the function PM_{sim} is that it has to be a monotonic increasing function, that is if $s_i \leq s'_i, i \in \{1, n\}$, then it is $PM_{sim}(s_1, \ldots, s_i, \ldots, s_n) \leq PM_{sim}(s_1, \ldots, s'_i, \ldots, s_n)$. This is intuitive, since better matches between base patterns can only increase the overall similarity score between corresponding complex patterns. Moreover, for the sake of simplicity, in the following we will assume that PM_{sim} is a commutative function. The optimal matching between regions, i.e. that for which $sim(cp_q, cp_s)$ is maximum, will be denoted as Γ_s^{opt}.

$$\begin{aligned}
sim(cp_q, cp_s) = \max\ & PM_{sim}(s_{i_1 j_1}, \ldots, s_{i_{|\mathcal{H}|} j_{|\mathcal{H}|}}), \\
& (i_h j_h), (i_l j_l) \in \mathcal{H}, (i_h j_h) \neq (i_l j_l) && (1) \\
\mathcal{H} = \{(i, j) &| x_{ij} = 1\} && (2) \\
\sum_{j=1}^{m} x_{ij} &\leq 1\ (i = 1, \ldots, n), && (3) \\
\sum_{i=1}^{n} x_{ij} &\leq 1\ (j = 1, \ldots, m), && (4) \\
x_{ij} &\in \{0, 1\}\ (i = 1, \ldots, n)(j = 1, \ldots, m) && (5)
\end{aligned}$$

Equation 1 means that to determine the overall score $sim(cp_q, cp_s)$ we have to consider only the matches $\Gamma_s()$ in $\mathcal{H}$ (Equation 2). Equation 3 (Equation 4) expresses the constraint that at most one pattern p_{s_j} of cp_s (resp. p_{q_i} of cp_q) can be assigned to a pattern p_{q_i} of cp_q (resp. p_{s_j} of cp_s).

Definition 1 (Correct matching). *A set of x_{ij} values that satisfies the constraints expressed by Equations 3, 4, and 5 is called a* correct *matching.*

Definition 2 (Complete matching). *A correct matching for which it is $\sum_{j=1}^{m} x_{ij} = 1, (i = 1, \ldots, n)$ (i.e. each pattern of cp_q is associated to a pattern of cp_s) is called a* complete *matching.*

[1] As an example, suppose that an user asks for an image containing two tigers: If a database image contains a single tiger, it is *not* correct to associate both query regions to the single “tiger” region of the DB image, since, in this case, information about the number of query regions is lost.

It should be noted that *any* correct matching for a pattern cp_s having a number of patterns lower than that of cp_q is obviously not complete.

Definition 3 (Optimal matching). *The correct matching that maximizes the function expressed by Equation 1 is called the* optimal *(or* exact*) matching, and will be denoted as* $\Gamma_s^{opt}()$*.*

5 Solving the Problem

A typical form of the scoring function PM_{sim} is that of a sum (this is indeed the case, save for a constant scale factor, for the image retrieval systems WALRUS [12] and WINDSURF [1]), leading to a re-formulation of Equation 1 as follows:

$$sim(cp_q, cp_s) = \max \sum_{i=1}^{n} \sum_{j=1}^{m} s_{ij} \cdot x_{ij} \tag{6}$$

The generalized assignment problem, in this case, takes the form of the well known Assignment Problem (AP), one of the most popular topics in combinatorial optimization. To resolve it, we can apply the Hungarian Algorithm [11] to the matrix $\{s_{ij}\}$ of similarity scores between regions. Sequential evaluation of a best matches query is performed by way of a simple algorithm that computes the optimal matching between the query pattern and all the searched patterns and returns the k patterns for which the highest similarity score is obtained [3]. Of course, the sequential algorithm requires to compute the similarity between the base patterns of the query and all the indexed base patterns. Moreover, the matching problem has to be solved for all the searched complex patterns.

In order to obtain a complexity sub-linear in the data set size, we describe an index-based algorithm that speeds up the evaluation of best matches queries by reducing the number of *candidate patterns*, i.e. patterns on which the optimal region matching problem has to be solved.

In order to use an index to speed-up the search, we suppose that the similarity between base patterns is computed by way of a distance between pattern features (this is the case, for example, for most of the region-based CBIR systems, see Section 3). In this case, a distance-based access method (DBAM), like the M-tree [6], can be used to index base patterns according to their respective (dis-)similarity. Such index structures are able to efficiently answer k nearest neighbor queries, as well as to perform a *sorted access* to the data, i.e. to output objects one by one in increasing order of distance with respect to a query [10].

To retrieve best matches for query patterns, we run a sorted access to the indexed patterns for each base pattern in the query. The $\mathcal{A}_0^{WS}$ algorithm shown in Figure 2 is able to return the correct result for a best matches query by only solving the matching problem for the *candidate set*, i.e. for those patterns having at least one base pattern that has been returned by a sorted access [3, 2]. The *random access* phase consists in computing those similarity scores s_{ij} between query patterns and patterns of candidates not returned in the X^i result sets.

Correctness of $\mathcal{A}_0^{WS}$ (the proof can be found in [3]) is independent of the specific PM_{sim} function used to combine scores into similarity between patterns, since it only relies on the monotonicity of PM_{sim}.

It can be noted that sorted and random access phases of $\mathcal{A}_0^{WS}$ somewhat resemble those of Fagin's $\mathcal{A}_0$ algorithm [8], the major difference being that $\mathcal{A}_0$ does not deal with the issue of correct matching, thus, if applied, it could report non-correct results.

6 Experimental results

Preliminary experimentation of proposed techniques has been performed on the WINDSURF system, using a sample medium-size data set consisting of about 2000 real-life images from the *IMSI-PHOTOS* CD-ROM.[2] The over 8000 obtained regions were indexed using an M-tree [6]. The query

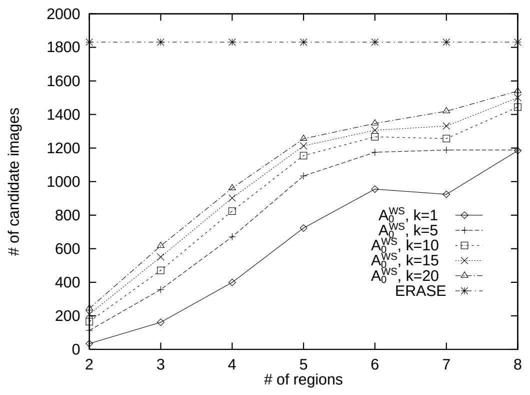
[2] IMSI MasterPhotos 50,000: `http://www.imsisoft.com`.

```
$\mathcal{A}_0^{WS}$($cp_q$: query, $k$: integer, $\mathcal{T}$: DBAM)
{ $\forall$ base pattern $p_{q_i}$ of $cp_q$, open a sorted access index scan on $\mathcal{T}$
    and insert results in the set $X^i$;
  stop the sorted accesses when there are at least $k$ patterns for which
    a complete assignment exists, considering only base patterns in $\cup_i X^i$;
  $\forall$ pattern $cp_s$ having base patterns in $\cup_i X^i$,
    $\forall$ pair $p_{q_i}, p_{s_j}$
      if $p_{s_j} \notin X^i$ compute score $s_{ij} = sim_{base}(p_{q_i}, p_{s_j})$; (random access)
    compute the optimal assignment; (combining phase)
  return the $k$ patterns having the highest overall scores $sim(cp_q, cp_s)$; }
```

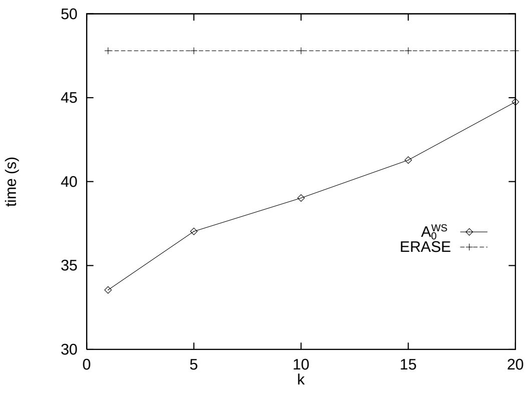
Fig. 2. The $\mathcal{A}_0^{WS}$ algorithm.

workload consists in about one hundred randomly chosen images not included in the data set. All experiments were performed on a Pentium II 450 MHz PC with 64MB of main memory running Windows NT 4.0.

The experiments we present concern the efficiency of the $\mathcal{A}_0^{WS}$ index-based algorithm as compared to that of the sequential algorithm. In Figure 3 (a) we compare the average number of candidate images, i.e. the images on which the Hungarian algorithm has to be applied, as a function of the number of query regions, for different values of k. Of course, the sequential algorithm (the horizontal line labeled ERASE, for Exact Region Assignment SEquential algorithm [3]) would lead to a number of candidate images equal to the number of images in the data set, whereas for $\mathcal{A}_0^{WS}$ this number depends both on k and on the number of query regions. As the graph shows, $\mathcal{A}_0^{WS}$ does well in reducing the number of candidate images. Clearly, its performance degrades as the number n of query regions increases, since the complexity of finding k objects in the intersection of n sets augments with n. This is also confirmed by Figure 3 (b), where query response times are shown for the case $n = 3$.



(a)



(b)

Fig. 3. Average number of candidate images vs. number of query regions (a), and response time vs. k ($n = 3$) (b).

7 Conclusions

In this work we have investigated the problem of correct resolution of best matches queries for complex patterns, obtained as sets of base patterns. In particular, an index-based algorithm ($\mathcal{A}_0^{WS}$) has been presented which computes the optimal matching between the base patterns, in order to maximize the overall similarity score between complex patterns, under the condition that only one-to-one matches exist. Preliminary experiments conducted over a region-based image retrieval system have shown that our approach is indeed very effective with respect to alternative retrieval strategies. In the future we plan to investigate how to solve the problem when different kind of constraints or aggregation logics exist, and also to devise algorithms for dealing with multiple levels of aggregation (i.e. when the composition hierarchy has more than just one level, as was the case considered in this work).

References

1. Stefania Ardizzoni, Ilaria Bartolini, and Marco Patella. Windsurf: Region-based image retrieval using wavelets. In *Proceedings of the 1st International Workshop on Similarity Search (IWOSS'99)*, pages 167–173, Florence, Italy, September 1999. IEEE Computer Society.
2. Ilaria Bartolini, Paolo Ciaccia, and Marco Patella. A sound algorithm for region-based image retrieval using an index. In *Proceedings of the 4th International Workshop on Query Processing and Multimedia Issues in Distributed Systems (QPMIDS 2000)*, pages 930–934, London/Greenwich, UK, September 2000.
3. Ilaria Bartolini and Marco Patella. Correct and efficient evaluation of region-based image search. In *Atti dell'Ottavo Convegno Nazionale SEBD*, pages 289–302, L'Aquila, Italy, June 2000.
4. Elisa Bertino, Barbara Catania, Matteo Golfarelli, Stefano Rizzi, Manolis Terrovitis, Panos Vassiliadis, and Michalis Vazirgiannis. A preliminary proposal for the panda logical model. Technical Report PANDA-UNIMI-2003-001, The PANDA Consortium, February 2003.
5. Chad Carson, Megan Thomas, Serge Belongie, Joseph M. Hellerstein, and Jitendra Malik. Blobworld: A system for region-based image indexing and retrieval. In *Proceedings of the 3rd International Conference on Visual Information Systems VISUAL'99*, pages 509–516, Amsterdam, The Netherlands, June 1999. `http://elib.cs.berkeley.edu/photos/blobworld/`.
6. Paolo Ciaccia, Marco Patella, and Pavel Zezula. M-tree: An efficient access method for similarity search in metric spaces. In *Proceedings of the 23rd International Conference on Very Large Data Bases (VLDB'97)*, pages 426–435, Athens, Greece, August 1997. Morgan Kaufmann.
7. Ingrid Daubechies. *Ten Lectures on Wavelets*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, 1992.
8. Ronald Fagin. Combining fuzzy information from multiple systems. In *Proceedings of the 15th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS'96)*, pages 216–226, Montreal, Canada, June 1996. ACM Press.
9. Venkatesh Ganti, Johannes Gehrke, Raghu Ramakrishnan, and Wei-Yin Loh. A framework for measuring changes in data characteristics. In *Proceedings of the 18th ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS'99)*, pages 126–137, Philadelphia, PA, May 1999. ACM Press.
10. Gísli R. Hjaltason and Hanan Samet. Distance browsing in spatial databases. *ACM Transactions on Database Systems*, 24(2):265–318, June 1999.
11. Harold W. Kuhn. The hungarian method for the assignment problem. *Naval Research Logistic Quarterly*, 2:83–97, 1955.
12. Apostol Natsev, Rajeev Rastogi, and Kyuseok Shim. WALRUS: A similarity retrieval algorithm for image databases. In *Proceedings 1999 ACM SIGMOD International Conference on Management of Data*, pages 396–405, Philadelphia, PA, June 1999. ACM Press.
13. Stefano Rizzi, Barbara Catania, Matteo Golfarelli, Maria Halkidi, Manolis Terrovitis, Panos Vassiliadis, Michalis Vazirgiannis, and Euripides Vrachnos. Towards a logical model for patterns. Submitted, 2003.
14. James Ze Wang, Jia Li, and Gio Wiederhold. SIMPLIcity: Semantics-sensitive Integrated Matching for Picture LIbraries. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(9):947–963, September 2001. `http://wang.ist.psu.edu/cgi-bin/zwang/regionsearch_show.cgi`.

Pattern Visualization by Pixel Validity Plots

Amihood Amir[1], Reuven Kashi[1], Daniel Keim[2], and Markus Wawryniuk[2]

[1] Department of Computer Science, Bar-Ilan University, Israel
[2] Department of Computer and Information Science, University of Konstanz, Germany

Abstract. The staggering amounts of available digital information, from astronomical, biological and scientific data to business and entertainment, create new challenges and new frontiers. To be fully useful, the data needs to be categorized and analyzed. Much effort has been devoted in recent years to develop automated methods for data analysis.
Because of the, somewhat amorphous, nature of clusters, visual methods have proven to be quite successful. However, because of the size and dimensionality of the data sets, it is necessary to develop automated clustering algorithms. Automated methods have the advantage of speed but the disadvantage of lacking domain knowledge.
We are proposing a novel method that makes it possible to automatically analyze high dimensional data with arbitrary clusters and high noise levels. Our method requires no domain knowledge in advance, yet it discovers different types of projected clusters and allows separating overlapping clusters with different topologies. At the core of our method lies the idea of subspace validity. We map the data in a way that allows us to test the quality of subspaces using statistical tests. Our approach is robust as far as noise is concerned, and enables discovery of arbitrary projected clusters with unusual topology even without any prior domain knowledge. Experimental results, both on synthetic and real data sets, demonstrate the potential of our method and show highly promising results.

1 Introduction

1.1 Motivation

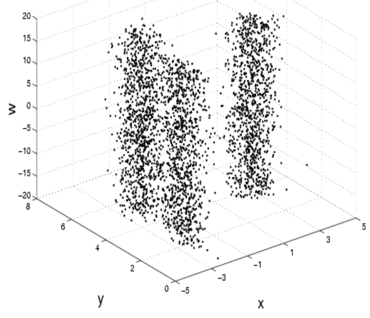
The last few decades witnessed a flood of digitized information, from pervasive digital libraries to an everexpanding Internet. Google reports over 3 billion sites, double the amount of four years ago.

The staggering amounts of available digital information, from astronomical, biological and scientific data to business and entertainment, creates new challenges and new frontiers. To be fully useful, the data needs to be categorized and analyzed. Much effort has been devoted in recent years to data analysis, borrowing from areas of machine intelligence, visualization, and statistics. These efforts produced myriad interactive and automated methods for data analysis.

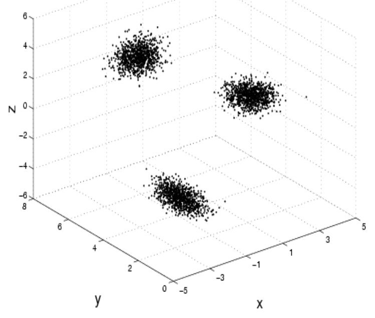
The concept of "cluster" is somewhat elusive. From an intuitive sense, it means points that are "close" to each other in space while they are "far" from other points. The meaning of such "closeness", vis-a-vis data analysis, is that points in a cluster are similar to each other, whereas points from different clusters are dissimilar.

This mapping from clusters to related variables over domain subsets adds some complications to our cluster definitions.

1. *Projected Clusters*: Typically, a relation may exist between some, but not all, variables. Consequently, the clusters are not defined over all attributes, i.e. values in some attributes are similar, but in other attributes not. This means that even finding all clusters in full space will not be sufficient. It is possible that projecting the space into a smaller dimensional space will yield interesting clusters that do not exist in the original data space. See example in Figure 1.
2. *Topology*: Clusters may have different shapes and linear dependencies. For example, we would like to identify a a sphere as a cluster, or we may be interested in identifying a plane in a multidimensional space as a cluster.
3. *Overlaps*: It is possible that under a certain projection, two clusters overlap and can not be distinguished, whereas in another projection they are separated. In an even more drastic situation, a cluster may not be identified under *any* (axes parallel) projection! See Figure 2.

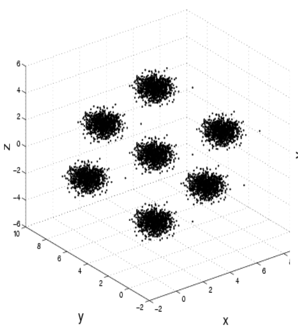


(a)
The clusters are not defined in dimension W

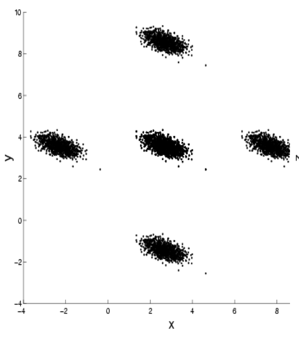


(b)
The clusters are defined in dimensions X,Y and Z

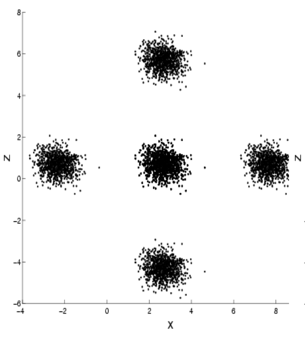
Fig. 1. Three 3-dimensional projected clusters in 4-dimensional space



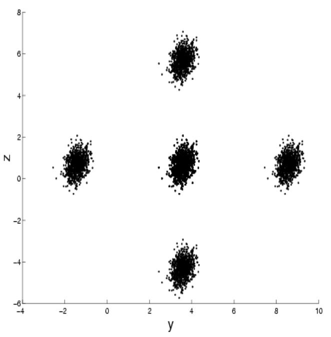
(a) A "surrounded" cluster



(b) The X-Y projection



(c) The X-Z projection



(d) The Y-Z projection

Fig. 2. A "hidden" cluster that does not appear in any axes-parallel projection

Because of the, somewhat amorphous, nature of clusters, visual methods have proven to be quite successful. Such methods use the perceptual capabilities of the human. Knowledge about the domain/task flows into the process step by step and is exploited both for a successful understanding of dependencies in the particular domain and for ferreting out and separating clusters [10, 11].

Albeit the success of such visualization methods, it is necessary to develop automated clustering algorithms. The reason for this need is the existence of high dimensional data sets that need to be analyzed. Just considering all different 2-dimensional projections of a d-dimensional dataset means analyzing $\binom{d}{2}$ cases. This is not a feasible number for interactive human analysis.

Automated methods have the advantage of speed but the disadvantage of lacking domain knowledge. Harnessing such knowledge through preprocessing work or by machine learning methods is a laborious process and the state-of-the-art is far from satisfactory. The above mentioned challenges of cluster finding do not have good solutions by current methods of automatic data exploration.

1.2 Related Work

Cluster finding has been an extensively studied problem for many years by the statistics, machine learning and database communities. In its *full-dimensional* view, the clustering problem may be defined as the problem of partitioning the set of data vectors into a number of clusters and noise, such that the data vectors within the clusters are *similar* to each other and the data items which are in different clusters or in the noise partition are *not similar*. Many specific clustering algorithms have been proposed [5, 9, 13, 6, 17] and to improve performance optimized clustering techniques have emerged [15, 18, 16, 7]. Recent research has proposed many algorithms for clustering [12]. However, in high-dimensional data sets, the curse of dimensionality severely affects the effectiveness (or quality) of the resulted clustering. It was shown in [8] that that existing clustering methods suffer from either severe breakdown in efficiency or have a serious effectiveness problem.

One of the first algorithms dealing with projected clustering is CLIQUE [3]. The algorithm mines the projection space bottom up by searching quantitative frequent item sets (histogram bins) which are assembled to clusters on a single linkage basis.

The algorithms PROCLUS [1] and ORCLUS [2] are k-means like algorithms, which need the number of clusters and the average dimensionality of the associated projections as parameters. Each cluster found is described by a single centroid from the data space with a set of vectors, spanning the subspace of the projected cluster. The data points are assigned to the centroids using a modified nearest neighbor rule. Both algorithms iteratively change the centroids and the subspaces, and stop after a predefined number of iterations. The iteration starts with random centroids, each associated with the full-dimensional space as cluster space. After assigning data points to the centroids the dimensionality for the subgroups is reduced. PROCLUS reduces the full-dimensional data space to the subspace spanned by the dimensions with the smallest variance. For the selection, the dimensions are treated independently, with the result that only axes-parallel projected clusters can be found. In contrast, ORCLUS determines for each cluster the eigenvectors of the covariance matrix with the smallest eigenvalues and therefore allows arbitrary orientations.

The most recent method DOC [14] defines a projected cluster as a hyperbox, with a boundary size of w (which is a parameter of the algorithm) in the bounded dimensions and an unbounded size in the other dimensions. Additionally, the number of data points in a projected cluster has to be at least a given minimum percentage of the total number of data points. An optimal projected cluster maximizes the number of bounded dimensions as well as the number of points in the cluster. DOC uses sampling to center the boxes around some randomly chosen data points. The result of the DOC method is a set of hyperboxes, which are bound in some dimensions and contain the projected clusters.

1.3 Our contribution

The described current methods are restricted to specific kinds of cluster definitions and cannot find in the data general structures that may have different topologies, share dimensions, and overlap. Such structures may still be considered as useful and interesting to the end user. The state-of-the-art methods for cluster finding differ in their requirements of knowledge domain and they require parameters (such as the requested number of clusters) as input for the algorithm. In addition, they depend on the amount of noise in the data and that affects the quality of the results.

Our goal is analyzing high-dimensional data sets in order to find structures and patterns which can be considered as interesting to the end user. It is accepted that if the human eye would perceptually capture a pattern in a subset of data points, then it is considered as valuable information which should be noticed and investigated further. The traditional way to capture such “similar” data objects is by the various definitions of clustering in the literature. Therefore, to demonstrate the performance – meaning the efficiency and the effectiveness – of the proposed methods we compare it against clustering algorithms in the databases literature. However, it should be stressed that we are not defining “clusters” in any traditional formal sense. We are seeking a more general method that can automatically detect “interesting” structures in high dimensional data sets. Therefore, our use of the word “clusters” to define such structures is intentionally quite loose.

We are proposing a novel method that makes it possible to analyze high dimensional data with high noise levels. Our method requires no domain knowledge in advance, yet it discovers projected clusters and allows separating overlapping clusters with different topologies.

At the core of our method lies the idea of *subspace validity.* We map the data in a way that allows us to test the parameters of a one-dimensional subspace. It is possible to perform various statistical tests efficiently in one dimension. We use a concept of *generalized histogram* to efficiently consider projections of three dimensions. In these projections, one of the variables is designated as

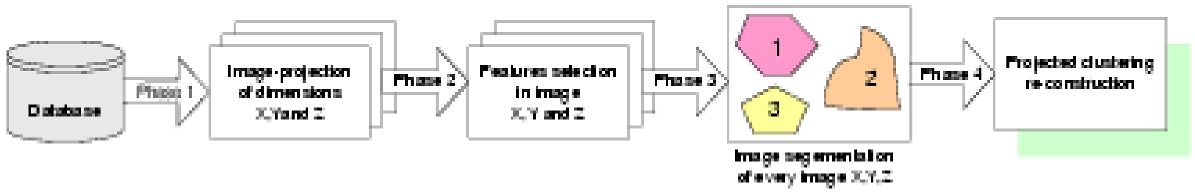

Database
Phase 1
Image-projection of dimensions X,Y and Z
Phase 2
Features selection in image X,Y and Z
Phase 3
1
2
3
Image segmentation of every image X,Y,Z
Phase 4
Projected clustering re-construction

Fig. 3. The proposed methodology

the subspace whose validity is checked by various statistical means. The result of these tests allows us to reach a conclusion about clustering in the three-dimensional subspace.

This approach is robust as far as noise is concerned, and enables discovery of projected clusters with unusual topology even without any prior domain knowledge. The main objective of this paper is introducing this new approach and demonstrating its effectiveness and viability.

2 The Concept of Subspace Validity

2.1 Problem Statement

The projected clustering problem consists of two main tasks, namely finding useful projections of the high dimensional data and determining clusters in these projections. Both tasks depend on each other, because a projection is only useful if it provides a good clustering and for the determination of a projected cluster a useful projection is needed. So an algorithm for the projected clustering problem has to decide, whether a clustering is good for a given projection, and it has to search the space of projections based on that criterion.

Given a database DB with m attributes, also referred as dimensions or variables, the goal is to find different local structures in DB, $C_1, \ldots, C_k$ with arbitrary topology, which can be defined over different subsets of dimensions. We refer to such local structure as *projected clusters* although we generalize our goal to general structures without any specific definition for similarity of objects. We define *(projected) clusters* as sets of data points that create a visually related structure.

Note that each cluster may be defined in a different projection and therefore, determining all projected density functions is computationally infeasible.

2.2 Overview of the Idea

Practical approaches to determining projected clusters are based on an iterative partitioning of the data. Partitioning based approaches try to partition the data space without partitioning any of the clusters. The most efficient way to partition the data set is to use low-dimensional projections to split the data. If we just use one-dimensional projections, the resulting clusters are hyperboxes. If we use two dimensional projections, we may separate arbitrary shaped regions of the data. In both cases, the idea is to split the data set only in the considered projection, which allows us to find projected clusters, however two dimensional projections have the potential of a more refined separation.

Our method is based on a new approach of automatically viewing and analyzing dataset projections by a method called *subspace validity* projections. Here we conceptually demonstrate our idea, a detailed description is given later.

Let DB be the database with m numeric attributes. We exploit the fact that if there exists a projected cluster in k dimensions, there is also a projected cluster in the subsets of the given k dimensions. The idea is to consider projected clusters in small sets of dimensions, call such a small set of dimensions a *projection set*. The problem with such a projection is that different clusters will tend to aggregate together and noise will have a much more prominent effect when we view clusters projected into a very small set of dimensions.

We propose to solve this problem with the concept of *subspace validity*. The idea is to consider an additional dimension, henceforth the *vertical dimension* to that of the small projection set. The vertical dimension is a given variable. We consider the behavior of that variable in every location of the space of the projection set. Every such location will, in effect, create a histogram in the vertical dimension. A contiguous subspace with "similar" histograms in the vertical dimension indicates a higher dimensional projected cluster.

Our projecting method is not restricted in the number of dimensions but for ease of exposition and to afford a more natural way of visualizing the idea consider three arbitrary attributes X, Y and Z. The reason that restriction to three variables aids in conceptualizing the idea is that the subspaces then map as images. Thus we can both view them and use customary digital image processing tools. It turns out, in addition, that the results obtained with this strategy are highly satisfactory. Consequently, the idea we pursue is considering different triples of variables $\langle X, Y, Z \rangle$ and automatically checking whether the *subspace validity* of the Z-variable in the $\langle X, Y \rangle$ subspace can indicates the existence of projected clusters in the X and Y variables. In other words, we will try to find triples of attributes $\langle X, Y, Z \rangle$, where the Z variable has values that belongs to some projected cluster(s) in meaningful regions of the X and Y variables.

The proposed approach has four major phases, as shown in Figure 3: The first stage is projecting the dataset on every subset $\langle X, Y, Z \rangle$ of three dimensions and constructing an "image" from it. The variables X and Y define the dimensions of the image matrix and the variable Z determines the information stored in each entry (i.e., pixel) of the image.

In the second stage, we map the data distribution of each pixel into features that allow us to test the similarity of contiguous pixels with respect to their designated one-dimensional vertical (i.e., the Z variable) distribution properties in the image. The result of this test enables us to find regions, referred to as *subspace validity* projections, in the image that indicate the existence of (projected) clusters in a three-dimensional subspace.

In order to discover such regions, we perform, in stage 3, a segmentation procedure on the images. The segmentation stage, discovers all regions of data points that have some interrelation in lower dimensions and enables us to efficiently focus on identifying the type of the relation towards clustering as shown in stage 4 of Figure 3.

In this final stage, we analyze the subspace validity regions obtained from all images to find clusters in subsets of dimensions greater than just three variables.

2.3 Conceptual Clusters and Subspace Validity

In this discussion we motivate our definition and analysis of the *subspace validity* concept. Subspace validity generalizes and formalizes the concept of *pixel validity* that was introduced in [4]. A pixel was called valid, if a large portion of its subpopulation is congregated within a small distance from the median of that subpopulation.

The typical shape of the density function of a one dimensional projection of a cluster is a unimodal distribution, and the Gaussian distribution is a good approximation. With the definition of the pixel validity using the median and the 50% rule a pixel with a Gaussian distribution in the vertical dimension values will be identified as valid. One can say that the vertical dimension values were identified as a projection of a cluster.

Let's assume now that we have a pixel and the distribution of the vertical dimension values is bimodal. With the original definition of pixel validity this pixel would be rated as invalid. But, if we consider that in reality clusters overlap or have peculiar shapes, an obvious approach is to test whether the two peaks in the density function correspond to two clusters.

A bimodal distribution of a histogram might indicate that points from its subpopulation come from two clusters. However, we must take care that two neighbored pixels, both with a bimodal distribution, belongs to the same two clusters, i.e. the two peaks of the bimodal distribution must have similar locations in the histogram. This is done by comparing histograms of neighboring pixels. This comparison necessitates a norm or distance function on histograms. In our implementation we use transforms for histogram comparisons.

The values of validity could be – uniform, unimodal (i.e. as in the original pixel validity definition), bimodal or multimodal. It is clear, though, that a region of similar histograms is interesting to the user. Thus it is necessary to find a good method of measuring distance between histograms, and a good method for separating the image into regions of similar histograms.

Let h be a data distribution (represented by a histogram or a density function). Let f, $f : h \rightarrow \overrightarrow{v}$, be the feature function of a data distribution h. The vector $\overrightarrow{v}$, also called the *feature vector*, is a d-dimensional vector, depending on the selected feature function.

For example, in the case of pixel validity definition, let p be some pixel. Then, h would be the histogram of the values that falls in the pixel p and the feature function f is the *median* of the values in p, i.e., $f(h) = median\{x : x \in p\}$. In this case the feature vector, $f(h)$, is a 1-dimensional vector. Other feature functions can be the Fourier transform (DFT) or Wavelet transforms, any statistic for testing k-modality of a distribution, or simply the histogram of a data distribution.

Given two feature vectors v_1 and v_2, we denote by s the similarity function such that $s(v_1, v_2)$ is the distance or the similarity between two feature vectors v_1 and v_2.

In general, we would like to define the feature vector as such that would best capture the distribution in the pixel and enable us to compare adjacent pixels for a similar regions or (projected) clusters.

3 Algorithms

3.1 Algorithms for Subspace Validity

In this subsection we explain in detail our subspace validity algorithms. We dwell specifically on the computation of various projections (images) from the original database, the extraction of features (associated with image pixels), and the analysis of regions meeting subspace validity.

Given an m-dimensional database DB, we first project the data set onto every subset of 3 dimensions. For each 3-dimensional projection, we designate one dimension (i.e., attribute) as the vertical dimension, i.e., the one-dimensional subspace whose subspace validity is tested.

Phase 1: Constructing compact images from the data. The first stage consists of building compact 2-dimensional projections or images for every triple of dimensions $\langle X, Y, Z \rangle$. A general framework of constructing gray level images from ordinary data was introduced in [4]. Let M_{XY} denote the matrix for attributes X and Y. In every M_{XY} entry we can store the values of the Z variable associated with that entry.

Note that the matrices we create are of relatively small size (e.g.,50×50). For this purpose we use a mapping $T : V \rightarrow C$, where V is the domain of variable values (e.g., X and Y), and $C = \{1, ..., c\}$, where c is some number, say 50. This mapping somewhat distorts the domain V but preserves distances in it, in the sense that a close clustering of many values will map to distinct numbers, whereas large empty areas will be clumped together. In a case where the V-values are (almost) uniformly distributed T can just be equally partitioning of the range of V into c bins.

Phase 2: Feature extraction from the image. Given three attributes X, Y, and Z, denote by M^Z_{XY} the corresponding image matrix, where X and Y are the coordinates and Z is the attribute which defines the values in each pixel. Recall that for the same (x, y) location in the image M^Z_{XY}, there can be many z values in the Z dimension. Therefore, in each pixel, we keep information about the data distribution of the Z values. This information is an example of a *feature vector* associated with the pixel. In general, feature extraction is carried out by applying some *feature function.*

Phase 3: Image segmentation. Since M^Z_{XY} represents an image, we can segment the image by applying standard *image segmentation* techniques. The segmentation of M^Z_{XY} yields regions, such that the pixels of a region have similar data distributions with respect to the Z variable. In other words, the image is segmented into regions such that a region's pixels have similar feature vectors.

Our objective is to identify, essentially, regions in a given image such that pixels in each region have similar features. We employ an image segmentation variant, such that segmented regions are dense, i.e., they contain a large number of points, and have compact shapes. The strategy we have used for segmentation is based on region growing, where at each iterative step pixels may be merged with a neighboring region depending on their feature similarity with that of the region.

Phase 4: Region analysis. At this stage we have possibly a large number of different regions of the various X and Y attributes, such that the values of the associated vertical Z dimensions in each such region are distributed in a similar manner. Using this information about the regions, we can analyze the data points in order to further discover prospective clusters (i.e., similar subsets of data points) either in the same attribute subspace or in some augmented subspace projections.

Note that each projected cluster may be defined with respect to a different projected subspace (i.e., attribute subset) and furthermore, clusters may overlap in some dimensions. According to our definition, (projected) clusters are subsets of data points. In order to determine the correct partitioning of data points into clusters we analyze the data distribution (or the density function) of the vertical dimension.

As discussed in Subsection 2.3, the data distribution of the vertical dimension in each region can suggest several possibilities regarding projected clusters. One simple case is a unimodal data distribution, which implies the existence of a single cluster in the appropriate dimensions. On the other hand, if the distribution is bimodal (or multimodal), we partition the data set over the vertical dimension and iterate the whole process with respect to this subset only (i.e., process each partition separately by again constructing "images" and finding regions).

The clustering scheme we have used consists of the following characteristics:

1. Consider the two-dimensional projection on X and Y. A pixel stores information about the distribution of the Z values of the points which are mapped to the corresponding cell.

2. We find regions (dense and compact) with similar distribution of Z.

3. If the distribution is unimodal (by definition of pixel validity), then these data points belong to one cluster in (at least) the X, Y, and Z dimensions. The data points should not be further partitioned with respect to the Z attribute in this case.

4. In order to augment the current set of dimensions which define the cluster (and to find clusters in the points which do not belong to the region) we partition all points into two sets; points which belong to the region and points which do not. The two subsets are processed recursively. The concept employed here is a generalization of the separator tree concept used in HD-Eye [10]. This method is explained in more detail in Section 3.2.

5. Suppose that a subset of points form a cluster in at least the set of dimensions D (e.g., $D = \{X, Y, Z\}$). Further suppose that these points yield a region of interest in the image corresponding to the triple U, V, W. The points of this region may thus define a cluster in the augmented set of dimensions $D \cup \{U, V, W\}$.

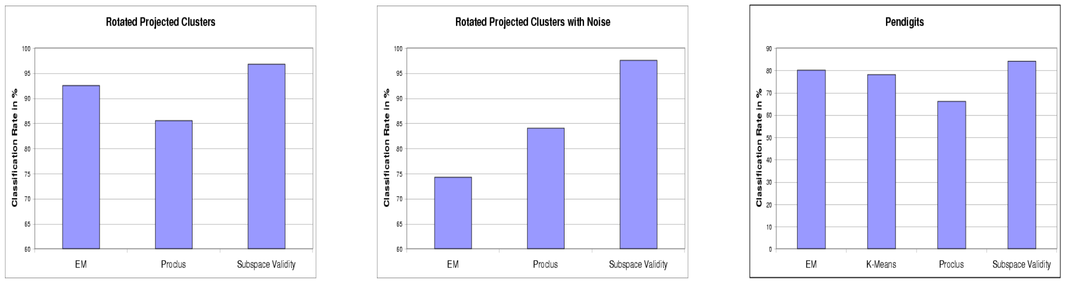
6. Searching for projected clusters is motivated by the fact that it might not be possible to augment a certain dimension set D (i.e., the cluster is not defined in all dimensions; it is a projected cluster).

3.2 Partition Tree Algorithms

The concept employed by our new framework for clustering is similar to that employed by decision trees to classification. A decision tree consists of a number of nodes, each containing a decision rule which splits the data to achieve finer labeling in the children nodes. In our trees, a node is a subspace of the dataset defined by a region that was discovered in a 2 dimensional projection by the segmentation algorithm. The decision rule is a separator defined by a region in a 2 dimensional projection of the points in this subspace.

A single region of similar features in a 2 dimensional projection does not necessarily define a single cluster in the dataset. The points defined by this region need to be further refined to understand the structure of clusters in that region.

This refinement is done recursively, by constructing all 2 dimensional projections of the subset of the dataset defined by the region. The order of these refinements is organized by the *partition trees.* A partition tree is a tree which corresponds to a recursive partitioning of a data set DB. A node v corresponds to a subset of DB. Like decision trees, which are an assemblage of classification rules forming a classifier, a partition tree is a collection of regions/separators forming a cluster model for the given data.



(a) synthetic rotated projected clusters

(b) synthetic rotated projected clusters with noise

(c) *pendigits* data set

Fig. 4. Experimental results

4 Evaluation and Comparison

We performed an extensive series of experiments to evaluate our new subspace validity method. In the experiments, we use a number of data sets with controlled characteristics, such as the number of dimensions or noise level, as well as real data sets. In the experiments, we compare our new approach to the most competitive existing methods including Expectation Maximization [5] and PROCLUS [1] (for a brief description of those algorithms see 1.2).

Real clusters often describe linear dependencies between the dimensions which define the cluster. This means that the cluster dimensions are not necessarily axis-aligned. Therefore, in our first experiment we analyze the ability of the algorithms to find rotated projected clusters. The clusters are modeled by multi-dimensional Gaussians and the rotation is defined by an arbitrary covariance matrix. We generate the clusters as follows: First, we create unrotated projected clusters. The cluster is an axis-aligned multi-dimensional Gaussian, but bounding dimensions (where points of a cluster are similar) have a small standard deviation in contrast to unbounded dimensions, where the standard deviation is very large. In a second step, we then apply a random rotation to the bounded dimensions.

The results of applying the different algorithms to this type of data sets are shown in Figure 4(a). The results represent the average classification rate over a number of data sets of this type. Subspace Validity has the best overall classification rate. EM performs surprisingly well and provides a better performance than PROCLUS.

In the next experiment, we examined the effect of noise in the data. For this series of experiments, we added random noise to the data sets from our second experiments and repeated the tests. The new results (see Figure 4(b)) show that the effectiveness of our method remains unchanged, but the performance of EM and PROCLUS degenerates. Interestingly, the performance of EM degenerates much faster than the performance of PROCLUS.

To show the relevance of our method, we performed a series of experiments with two real data sets. The first data set are the *pendigits* data set from the University of California at Irvine's Machine Learning Repository (see www.ics.uci.edu/$\sim$mlearn/MLRepository.html). The pendigits data set contains 7,494 tuples and 16 dimensions that describe handwritten digits. The data set is labeled, we are therefore able to determine the classification rate as in the experiments above.

The results of our experiments are shown in Figure 4(c). With our method we get the best classification rate of 84%, followed by EM with 80% and K-Means with 78%. From the bad classification rate of PROCLUS (only 66%), we may conclude that clusters are spread over all dimensions.

Figure 5 gives an exemplary pixel validity plot. The image has a resolution of 64×64, unoccupied pixels are colored with white, whereas pixels with similar features have the same color. In this case we identified 3 groups of similar features. Note, that in this case regions of similar features are not necessarily connected. But in order to process the image recursively, the algorithm will select connected regions only. To understand the meaning of the different regions, the average of the histograms of the third dimensions is shown for each group of similar features. Here, the average is shown with red bars. Because some histograms of a group might be outliers, the 75% quantile of the histograms is shown as well. In most cases the average and the quantile of a bin are very close. Therefore we can conclude that the regions represent different characteristics of the dataset in this particular projection with respect to particular the third dimensions.

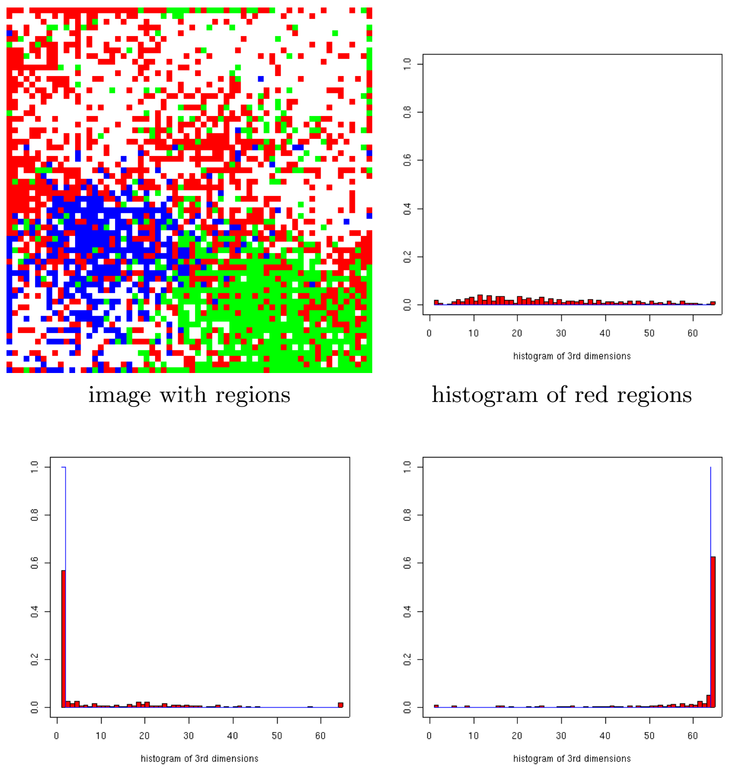


image with regions histogram of red regions

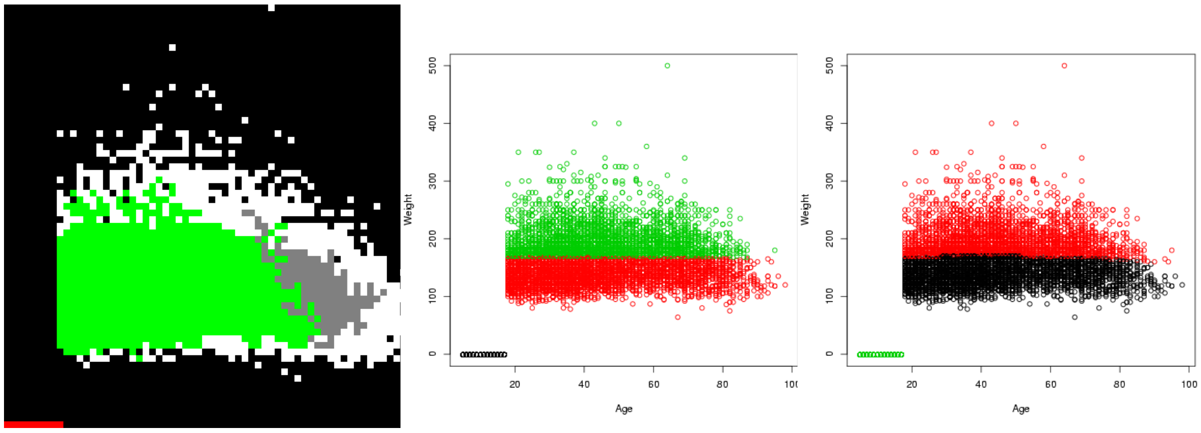
histogram of blue region histogram of green region

Fig. 5. Example of an segmentation

Our second real data set is the census dataset nhis93ac, National Health Interview Survey 1993 (NHIS). The data set is available from http://ferret.bls.census.gov/. It has several hundred numeric and nominal attributes and consists of 45951 records. We selected the most promising numerical attributes, which are *Person's age*, *Bed Days in Past Year*, *Doctor Visits in Past Year*, *Years of Education*, *Family Income Level*, *Number of Conditions*, and *Weight*.

Since the data does not come with a known classification, we can not calculate classification rates but have to judge the classification accuracy by looking at the results. For this purpose, we use the Subspace Validity plots and visualizations of the corresponding projections for the results of EM and K-Means.

An indicative example of the superiority of the subspace validity method can be seen in Figure 6. The clusters found by the EM method and the K-Means method are pretty much meaningless. However, the clusters found by our method tell a very interesting story. The horizontal axis is the age and the vertical axis is the weight, with the origin in the bottom left corner. Note the diagonal in the projected cluster, from top left to bottom right. This indicates something that insurance companies love to find. That the number of medical conditions increases at a younger age when the weight is greater. In fact, the diagonal can even predict at what weight and at what age one can expect the problems to accumulate.



(a) Result of Subspace Validity (b) Result of K-Means (c) Result of EM

Fig. 6. An example from census data set 'nhis93ac'

References

1. C. C. Aggarwal, C. M. Procopiuc, J. L. Wolf, P. S. Yu, and J. S. Park. Fast algorithms for projected clustering. In *Proc. ACM SIGMOD International Conference on Management of Data, June 1-3, 1999, Philadephia, Pennsylvania, USA*, pages 61–72. ACM Press, 1999.
2. C. C. Aggarwal and P. S. Yu. Finding generalized projected clusters in high dimensional spaces. In *Proc. of the 2000 ACM SIGMOD International Conference on Management of Data, May 16-18, 2000, Dallas, Texas, USA*, pages 70–81. ACM, 2000.
3. R. Aggrawal, J. Gehrke, D. Gunopulos, and P. Raghavan. Automatic subspace clustering of high dimensional data for data mining applications. In *Proc. ACM SIGMOD International Conference on Management of Data, June 2-4, 1998, Seattle, Washington, USA*, pages 94–105. ACM Press, 1998.
4. A. Amir, R. Kashi, and N. S. Netanyahu. Analyzing quantitative databases: Image is everything. In *VLDB 2001, Proc. of 27th International Conference on Very Large Data Bases*, pages 89–98, Roma, Italy, September 11-14 2001.
5. A. P. Dempster, N. Laird, and D. Rubin. Maximum likelihood for incomplete data via the EM algorithm. *J. of the Royal Statistical Society, ser. B*, 39:1–38, 1977.
6. M. Ester, H.-P. Kriegel, J. Sander, and X. Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proc. of the 2nd Intl Conf. on Knowledge Discovery and Data Mining*, pages 226–231. AAAI Press, 1996.
7. A. Hinneburg and D. A. Keim. An efficient approach to clustering in large multimedia databases with noise. In *Proc. of the 4th Int. Conf. on Knowledge Discovery and Data Mining*, pages 58–65. AAAI Press, 1998.
8. A. Hinneburg and D. A. Keim. Clustering methods for large databases: From the past to the future. In *Proc. ACM SIGMOD International Conference on Management of Data, June 1-3, 1999, Philadephia, Pennsylvania, USA*, page 509. ACM Press, 1999.
9. A. Hinneburg and D. A. Keim. Optimal grid-clustering: Towards breaking the curse of dimensionality in high-dimensional clustering. In *Proc. of 25th International Conference on Very Large Data Bases, September 7-10, 1999, Edinburgh, Scotland, UK*, pages 506–517. Morgan Kaufmann, 1999.
10. A. Hinneburg, M. Wawryniuk, and D. A. Keim. Hd-eye: Visual mining of high-dimensional data. *IEEE Computer Graphics & Applications Journal*, 19(5):22–31, September 1999.
11. D. A. Keim. Information visualization and visual data mining. *IEEE Transactions on Visualization and Computer Graphics (TVCG)*, 8(1):1–8, January–March 2002.
12. D. A. Keim and A. Hinneburg. Clustering techniques for large data sets - from the past to the future. In *Tutorial Notes for ACM SIGKDD 1999 International Conference on Knowledge Discovery and Data Mining*, pages 141–181, San Diego, CA, 1999. ACM Press.
13. R. T. Ng and J. Han. Efficient and effective clustering methods for spatial data mining. In *Proc. of 20th International Conference on Very Large Data Bases, September 12-15, 1994, Santiago de Chile, Chile*, pages 144–155. Morgan Kaufmann, 1994.

14. C. M. Procopiuc, M. Jones, P. K. Agarwal, and T. M. Murali. A monte carlo algorithm for fast projective clustering. In *Proc. of the ACM SIGMOD international conference on Management of data*, pages 418–427. ACM Press, 2002.
15. E. Schikuta. Grid-clustering: An efficient hierarchical clustering method for very large data sets. In *In Proc. 13th Int. Conf. on Pattern Recognition, volume 2*, pages 101–105, Vienna, Austria, October 1996. IEEE Computer Society Press.
16. W. Wang, J. Yang, and R. R. Muntz. Sting: A statistical information grid approach to spatial data mining. In *Proc. of 23rd International Conference on Very Large Data Bases, August 25-29, 1997, Athens, Greece*, pages 186–195. Morgan Kaufmann, 1997.
17. X. Xu, M. Ester, H.-P. Kriegel, and J. Sander. A distribution-based clustering algorithm for mining in large spatial databases. In *Proc. of the Fourteenth International Conference on Data Engineering, February 23-27, 1998, Orlando, Florida, USA*, pages 324–331. IEEE Computer Society, 1998.
18. T. Zhang, R. Ramakrishnan, and M. Livny. Birch: An efficient data clustering method for very large databases. In *Proc. of the 1996 ACM SIGMOD International Conference on Management of Data, Montreal, Quebec, Canada, June 4-6, 1996*, pages 103–114. ACM Press, 1996.

Using Pattern-base Management Systems – Requirements and Applications

Martin Nelke[1]

[1] MIT – Management Intelligenter Technologien GmbH
Pascalstrasse 69, D-52076 Aachen, Germany
Phone: +49-2408-94580, Fax: +49-2408-94582
martin.nelke@mitgmbh.de
http://www.mitgmbh.de/e/index.htm

Abstract. A successful pattern data base management system has great potential in applications in the area of Business, Trade, Services and Industry as we can see that in these areas we have a tremendous increase of data and very little offer from focused commercial products. One problem for market penetration is that the data mining community coming from an academic background has focused on the algorithms behind the technology. But efficient algorithms are not what business users care about. The core algorithms are now a small part of the overall application and users do care about compact and rigid information which a Pattern-base Management System (PBMS) could offer. Also the results of the data mining process or of a dbase query will drive efforts in areas such as marketing, risk management, and credit scoring. Each of these areas is influenced by financial considerations that need to be incorporated in the focused data analysis process. A business user is concerned with maximizing profit, not minimizing a function error and thus the necessary information to make these financial decisions (costs, expected revenue, etc.) is often available and ideally should be incorporated in the pattern. This paper deals with the requirements of building a PBMS and contains an overview on existing and future applications.

1 Introduction

Pattern is a compact and rich in semantics representation of raw data, with the main characteristic that it is a repeated attribute or constraint shared by the raw data that satisfy it. The design of a Pattern Based Management System, i.e. a system for handling (storing/processing/retrieving) patterns extracted from raw data in order to efficiently support pattern matching and to exploit pattern-related operations generating intentional information is the aim of the PANDA ("Patterns for Next-Generation Database Systems") IST/FET Working Group (IST-2001-33058).

PANDA concepts and techniques cover numerous application domains dealing with voluminous and heterogeneous data (further called raw data). Pattern extraction algorithms (pexas) applied to these data generate potentially large quantities of patterns that essentially represent the knowledge hidden in the raw data and due to their volume and diversity call for efficient management.

The requirements to design a PBMS from the application and users's view are discussed in the following .

2 Requirements

To support a successful use of a PBMS, the following requirements should be addressed:

- To support IT professionals to build patterns on users' data offering flexible interfaces for data access, parameter settings
- To reduce the dependency on IT professionals to apply data mining models on users' data.
- To reduce the cost of data analysis (storage capacities, performance of building patterns and applying models).

The advantages of a Pattern Based Management System designed in the PANDA working group should be pointed out clearly to give interested business users investment incentives.

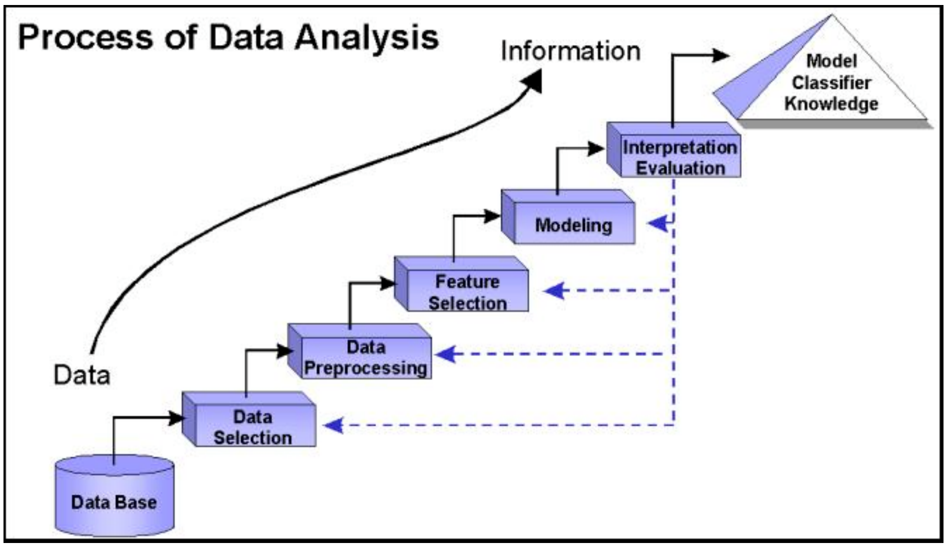


Fig. 1. Overview process of building patterns

From the requirements point of view to build patterns, a PBMS should support the whole data analysis process considering as main aspects the generation, storage, analysis, processing and retrieving of patterns [1].

It actually starts with the acquisition and storage of data in a data base; a flexible import and export interface is necessary for an easy data exchange.

Considering the vast amounts of data that can be acquired from business processes, the data to be analyzed has to be carefully selected. In the pre-processing stage the data is filtered, missing values and outliers are treated, derived quantities are calculated from the basic process variables and all the variables are trans-formed and normalized. This step is followed by a feature selection procedure that is supposed to ensure that only the most relevant of the pre-processed variables are used for modeling. In the modeling phase ("building patterns") data mining technologies are applied to construct a model (e.g. decision tree, cluster, association rule).

The efficient support of pattern matching and the exploitation of pattern-related operations are focused by the business end users. Their main requirements are

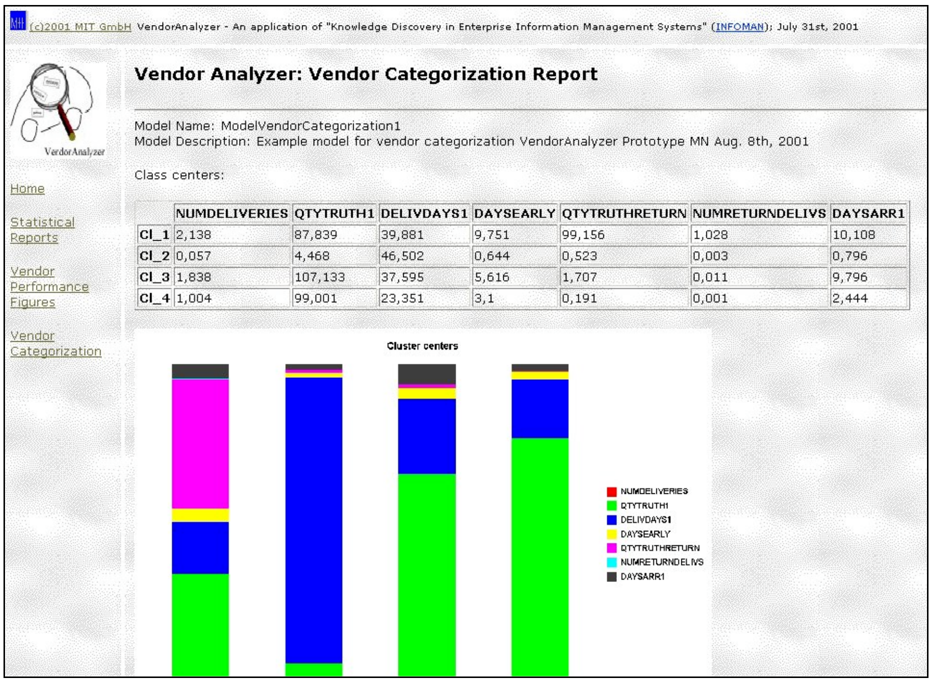
- Documentation: a user should be able to select the right pattern / model for his application supported by a report about the process of building the model (input raw data, parameters, evaluation results). This is necessary i.a. to follow rules and regulations for the use of software systems in companies e.g. of financial business
- Reporting: the results of building and retrieving of patterns should be summarized in reports
- User interface: a web-based graphical user interface should be supported to offer an easy access to the PBMS functions
- Historic data: it should be possible to follow the development of; e.g. if we analyze a customer of a financial institution by cluster analysis, his features like income, age, .. will change during in his lifetime. If a pattern-matching model is applied to such customers, it is very interesting to see to which patterns the customer belonged in the earlier years
- Methodology: it would be interesting not only to use patterns (e.g. in a matching application) in a crisp way (that means the decision of a pattern matches or not), but to give an additional information about the degree of the matching as possible with the Fuzzy C-Means classifier [2].
- Adaptivity: However, the need for adaptivity in real application problems is the same and the end-user is not technology dependent: he just wants his problem to be solved. A PBMS should consider the adaptation of the used technologies to changing circumstances regarding raw data, algorithms, parameters, … to optimize the PBMS performance

3 Applications

3.1 Supplier Relationship Management

Procurement already serves as an ideal platform to change a company's value proposition, influence the choice of core competencies to maintain internally, and enable its ability to innovate. By making the scope of supplier relationships broader and more flexible, companies are enabled to achieve new areas of growth and to build profitable new business models. Global leaders identify procurement excellence as the main source of sustainable competitive advantage.

One of the most typical applications of Enterprise Resource Planning (ERP) systems like SAP R/3® is being used for at end user sites is logistics [3],[4]. Here, vendor selection and evaluation is one of the important tasks in the supply chain management process. This is based on the two information sources of buying market research results and vendor controlling information. Buying market research typically collects, prepares and analyses vendor information on general company information, product based data, conditions and services, and purchaser own relationship to the vendor. Vendor controlling, on the other hand side, as based on information gathered from the company's own experience with the vendor, is the more reliable source.



(c)2001 MIT GmbH VendorAnalyzer - An application of "Knowledge Discovery in Enterprise Information Management Systems" (INFOMAN); July 31st, 2001

Vendor Analyzer: Vendor Categorization Report

Model Name: ModelVendorCategorization1
Model Description: Example model for vendor categorization VendorAnalyzer Prototype MN Aug. 8th, 2001

Class centers:

	NUMDELIVERIES	QTYTRUTH1	DELIVDAYS1	DAYSEARLY	QTYTRUTHRETURN	NUMRETURNDELIVS	DAYSARR1
Cl_1	2,138	87,839	39,881	9,751	99,156	1,028	10,108
Cl_2	0,057	4,468	46,502	0,644	0,523	0,003	0,796
Cl_3	1,838	107,133	37,595	5,616	1,707	0,011	9,796
Cl_4	1,004	99,001	23,351	3,1	0,191	0,001	2,444

Fig. 2. Screenshot typical patterns for vendor evaluation

Vendor evaluation can be defined as a permanent and objective monitoring and evaluation process of a vendor's performance regarding specific criteria like, e.g., quality or timeliness. The main objective of vendor evaluation is to ensure product and service quality and the optimization of the vendor group structure according to the business of the purchasing organization. This objective is highly correlated with constraints like, e.g., permanent cost reduction and maintenance of competitive advantage. Knowledge Discovery (KD) is an extraction process of implicit, unknown, potentially useful and understandable information from large data sets using data mining technologies. Vendor categorization by patterns that is based on a segmentation of the whole vendor portfolio helps to identify intra-segment similarities and inter-segment dissimilarities. This, in turn, helps to get a deeper understanding the vendor portfolio structure, and to optimize this portfolio according to the company's business strategy. Dynamic generated typical profiles of vendors described by patterns can identify trends at early stages and deliver important controlling information.

Typical raw data for vendor evaluation include

- dates and amounts of orders,
- dates and amounts of deliveries,
- quality and test results of deliveries,
- vendor information.

First approaches use features like number of deliveries, quantities, number of days between delivery and order and others as input for a cluster algorithm to identify patterns in the vendors performance to the requested deliveries [5],[7].

3.2 Customer Segmentation in Financial Services

Customer Relationship Management is the practice of identifying, attracting and retaining the “best” customers to increase sales and profits. To achieve this target several independent S/W providers and vendors developed stand alone or add-ons to provide solutions for Customer Relation Management.

But there is no product in the market that could successfully use and manage patterns that are concentrating compact information for the customer although most traditional data bases in the companies have only the basic raw data information. It is obvious that data like purchase patterns, current status, contact history, demographic information, sales results and service trends are there; but that data must also be "actionable," so that managers and employees on the front lines can use it for decision support. What is a great challenge in this area is "changing patterns": market and customers are moving fast so it is difficult to work and derive conclusions using static patterns from the past. Customer segmentation has its origin in the field of marketing and market research. It provides analytical division of all potential customers in a sales market according to different criteria. This results in the formation of internally homogeneous and externally heterogeneous groups of customers or customer segments, thus providing marketing activities focused on these segments.

The most important component for success in financial services is certainly the relationship between the respective institute and its customers. This relationship and the customer's degree of satisfaction and confidence is of central importance since it concerns the customer's property. It determines the strategic aim in banking and finance. The needs of individual customers are crucial for strategic planning in this area. In addition to the necessity of providing an enormous variety of products, most financial institutes serve a broad palette of customers. Due to this financial institutes have to adapt as fast as possible to changing requirements and structures in order to remain in a strong position among a rising number of competitors. Confronted with this, financial institutes have realized that a decisive approach for the improvement of quality in consultation and services lies in the segmentation of all customers into different target groups containing 'similar' customers.

The products and services for a certain customer segment must be provided in a way and at a level which the customer belonging to this segment is expecting and willing to pay for. The relevance of a customer-oriented segmentation and the identification of different customer segments by suitable methods is required for such an approach [6].

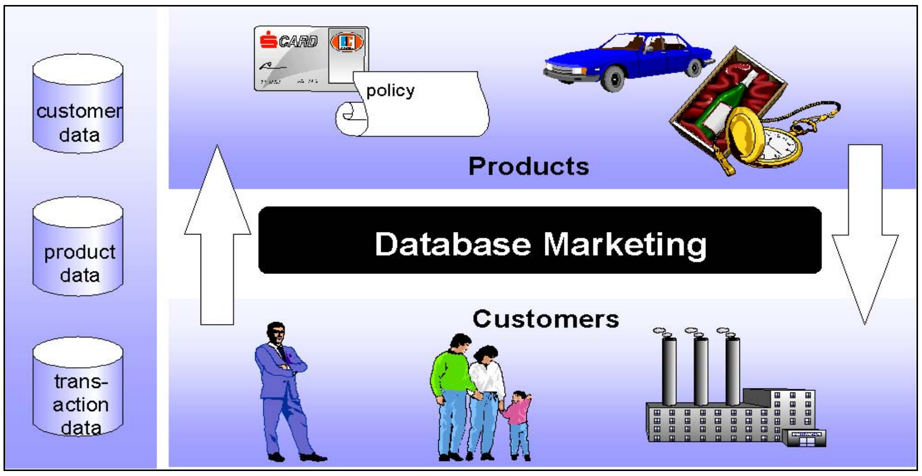


Fig. 3. Database marketing uses the matching of product to customers belonging to patterns generated from raw data

Cluster analysis is the usually used method for customer segmentation referring to a group of methods for recognition of structure in data sets. It can be applied if certain features of different objects (e.g. customers) can be observed, and if those objects are supposed to be divided according to their individual feature values. The entire set of objects will be clustered regarding all describing features in such a way that all objects belonging to one cluster represented by a pattern are (possibly) similar. Vice versa, the objects belonging to different clusters should be different regarding their feature values [8].

3.3 Future Applications

Future applications of a PBMS will have great success where useful information from lots of available raw data can be generated with high performance. Interesting domains in general for business success are security, health, communication, financial business, marketing and distribution of goods as well as entertainment.

Especially supported by the growing multimedia business with digital image processing and GSM/GPS in recent years applications like

- personal identification by pattern matching derived from visual information,
- traffic analysis by detection of cars on highways,
- customer profiling in using the internet

could be target of a PBMS approach. One of the great advantages of the use of patterns is the reduction of raw data to because of the representation aspect which hides the original identity when the raw data contains confidential information.

Regarding the aim of PANDA working group, the design of a PBMS as a result should be able to show the technical proof of concept as well as its advantages to estimate the benefit for end-users to have the chance of starting an implementation.

References

1. Angstenberger, Joachim; Weber, Richard; Poloni, Marco, 1998, "Data Warehouse Support to Data Mining: A Database Marketing Perspective ", The Journal of Data Warehousing, pp. 2-11
2. Bezdek, J. C., 1981, Pattern Recognition with Fuzzy Objective Function Algorithms. Plenum Press, New York
3. Grimmer, Udo; Poloni, Marco: “VendorAnalyzer: A real life vendor profiling tool - Data mining on top of mySAP.com” , European SAP-Microsoft Congress 2001, February / March 2001, Berlin; Germany
4. i2: “Supplier Relationship Management (SRM) - Powering the Bottom Line through Strategic Supplier Relationships”, WP-6709 (02/01), January 2001
5. Nelke, Martin: “Supplier Relationship Management: Advanced Vendor Evaluation for Enterprise Resource Planning Systems”, Proceedings of Eunite2001, December 13-14 2001, Tenerife, Spain
6. Nelke, Martin: Business Intelligence in Financial Engineering - A Combined Approach to Customer Segmentation and Database Marketing; Coil 2000: Symposium on Computational Intelligence and Learning; 22-23 June 2000, Chios / Greece
7. Nelke, Martin; Klotz, Uwe; Poloni, Marco: “A new Vendor Evaluation Product for SAP R/3® Systems”, Session “Knowledge Discovery in Enterprise Information Management SAP R/3 Systems” of European Symposium on Intelligent Techniques, September 14-15, 2000, Aachen, Germany
8. Zimmermann, H.-J., 1987, "Fuzzy Sets, Decision Making, and Expert Systems", Kluwer Academic Publishers, Boston, Dordrecht, Lancaster

Mining patent databases for monitoring technological trends

Konstantinos MARKELLOS[1,2], Penelope MARKELLOU[1,2], Giorgos MAVRITSAKIS[1,2], Katerina PERDIKOURI[1,2], Spiros SIRMAKESSIS[1,2], Athanasios TSAKALIDIS[1,2]

[1] Research Academic Computer Technology Institute,
61 Riga Feraiou Str., 26221 Patras, Greece,
[2] University of Patras, Computer Engineering and Informatics Department,
Campus, 26500 Patras, Greece
Tel: +30-2610-960335, Fax: +30-2610-960322
E-mail: {kmarkel, markel, syrma, tsak}@cti.gr
E-mail: {mayritsa, perdikur}@ceid.upatras.gr

Abstract. The analysis of the information "hidden" in patent databases can provide a very clear view of the current trends regarding technological and scientific innovation. In this paper we present the methodology used from a system that combines efficient and innovative tools for the analysis of textual data related to patents. The system uses existing patent databases as input, supports multidimensional analysis and produces new technological indicators as output. These indicators express information concerning the scientific and technological progress and can help the active actors (individuals or organizations) to understand the on-going changes and their effects.

1 Introduction

Nowadays in order to observe and analyse economic, technological or scientific activities it is necessary to take into account the flow of information that is related to them. Usually information is stored in large databases and therefore access to information is performed through access to those databases. Information can be distinguished in several categories e.g. information related to research is stored in research publications, scientific magazines etc., while information related to the phases of development and production is mainly stored in patents.

A patent is a legal title granting its holder the exclusive right to make use of an invention for a limited area and time by stopping others from, amongst other things, making, using or selling it without authorization [7]. Patents indicate the level of innovative activity in a particular market [10]. They generate new investment and are a motivating force behind technical progress. Patents are also closely related to technological and scientific activities [4] i.e. with Technology Watch. Therefore it is closely related to innovation. Statistical exploitation of this information may lead to useful conclusions related to technological development, trends of technology or innovation.

A patent can be decomposed and described by several fields. Each field contains specific information while each patent is described by a code (or in many cases more than one codes) depicting its technical characteristics. The information included in patents can be categorized in four large sections as shown in figure 1. The first section concerns the technological features of each invention, the second section refers to all the characteristics concerning the application of each invention. In addition information about the countries in which an invention is protected is provided as well as information about the invention's origin. Below are presented the fields describing a patent in the database ESPACE ACCESS:

- PN: Priority Number (number of the patent).
- AN: Application Number.
- PR: Priority Year.
- DS: Designated States.
- MC: Main Classification Codes.
- IC: All Classification.
- ET: English Title.
- FT: French Title.
- IN: Inventor.

- PA: Applicant (name of the company depositor).
- AB: English Abstract.
- AF: French Abstract.

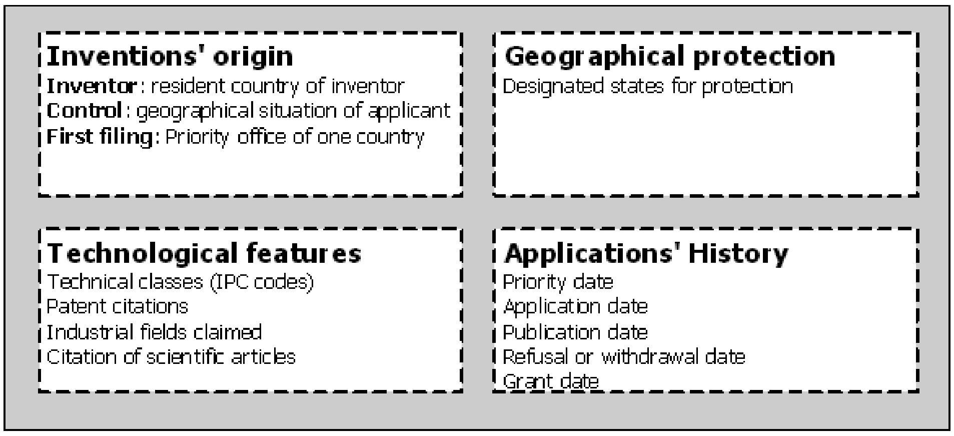

Inventions' origin
Inventor: resident country of inventor
Control: geographical situation of applicant
First filing: Priority office of one country
Geographical protection
Designated states for protection
Technological features
Technical classes (IPC codes)
Patent citations
Industrial fields claimed
Citation of scientific articles
Applications' History
Priority date
Application date
Publication date
Refusal or withdrawal date
Grant date

Figure 1. Information contained in a patent

The mining and the analysis of the information “hidden” in patents, which are stored in many international databases, can provide a very clear view of the current trends regarding technological and scientific innovation and can provide measurable and comparable results [6], [11].

In this paper we present a system that combines efficient and innovative methodologies and tools for the analysis of textual data related to patents. The system uses existing patent databases (input), supports multidimensional analysis and produces new indicators (output). These indicators express information concerning the scientific and technological progress andcan help the active actors (individuals or organizations) to understand the on-going changes and their effects.

The structure of this paper is the following. In section 2 we discuss our system for the analysis of patents and present its functional architecture. Section 3 describes the adopted methodology. System’s evaluation and a list with its key benefits are included in section 4. Finally, in section 5 some conclusions are summarised.

2 The system

The implemented system comprises a flexible tool, specialized in the mining and the analysis of patent data. It combines different features and technologies as well as a complex statistical methodology for analysing huge amounts of textual data. Specifically, it is characterized by its flexibility and interactivity as well its ability to produce results in a fast and accurate way. The value of this system is enhanced by the fact that there are available different ways of visualization. The user is able to visualize the results through graphs easily adapted to his own requirements, tables and ready-made reports produced through a drag and drop procedure.

The functionalities of the system, the analyses offered as well as the visualization techniques are in total agreement with User Requirements analyses. In fact the design of the functionalities as well as of the analyses offered were based on the feedback obtained from the questionnaire survey in many companies at national and international level that are specialized in the domain of patent management as well as on the feedback obtained from the user group meetings.

The system is windows-based with a user-friendly interface (see figures 3, 4, 5, 6, 7, 8, 9, 10, 11 in the Appendix). It is a modular system consisting of different independent modules giving to the user the ability to perform various types of analysis adapted to the user needs. Each module corresponds to a main system task, has a pre-specified input and output data format and executes a set of well-defined tasks. The main systems modules are:

- **Database Manager Module:** performs the data import, cleaningand preparation. Specifically, the module reads the data from the selected patent database and transforms them into the appropriate format, in order to be ready for further processing. The system accepts the textual and numerical data of the patents fields.

- **Statistical Analysis Module:** performs the simple statistical analysis, as well as the analysis envisaged from the proposed methodology. In particular, it applies textual analysis methods on the pre-formatted data, in order to extract valuable information and create the first groups of patents.
- **Results Presentation Module:** performs the data export and the graphically representation of the results. It is very important for the end-user in order to fully understand the meaning of the produced results.

Next figure presents the system's functional architecture. The depicted flows define the different stages of the analysis and the connections between them. It is also common sense that the natural sequence followed by the flows should be respected for the correct operation and for the robustness of the results.

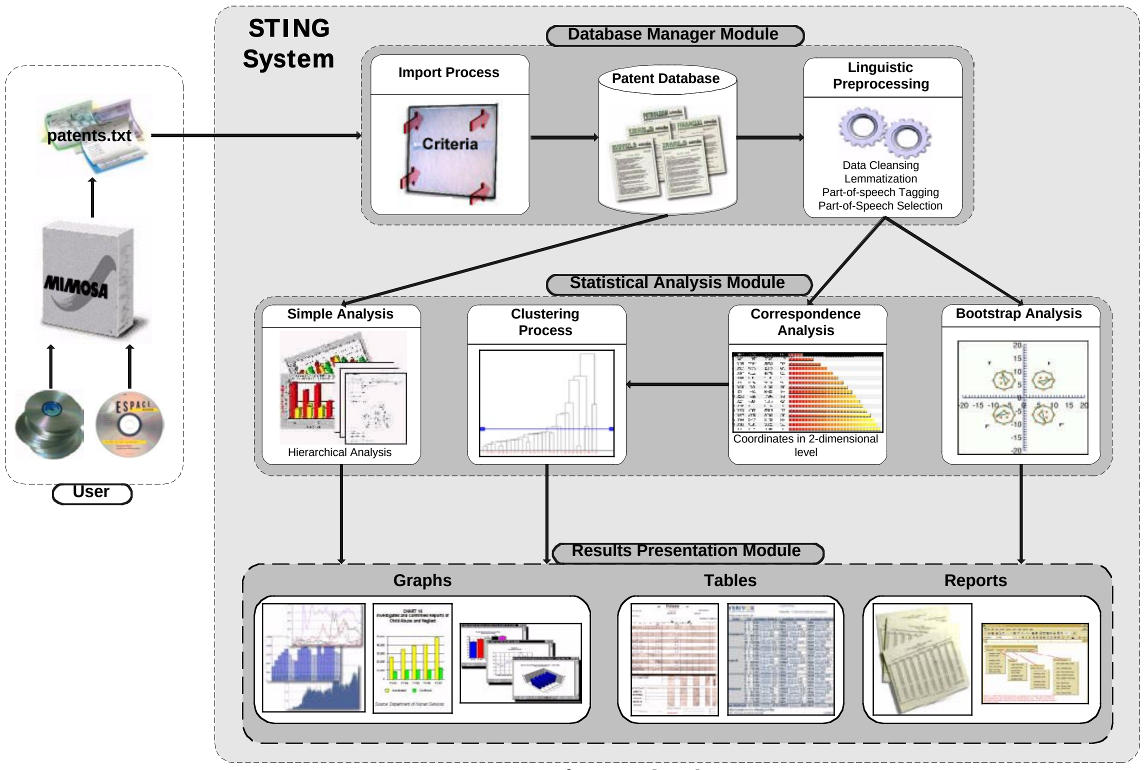


Figure 2. System's functional architecture

3 The methodology

The system tools interact and allow easy navigation to the user from the Data import level to the Statistical Analysis module and the Results Presentation Module. The system uses innovative IT technologies and complex statistical methodology. This methodology is based on the use of textual analysis [2] and complex statistical procedures:

- **Textual analysis** consists in:
  - *Data cleansing:* cleans the input data by removing irrelevant html characters and punctuation characters.
  - *Lemmatisation:* focuses in restricting the morphologic variation of the textual data by reducing each of the different inflections of a given word form to a unique canonical representation (or lemma).
  - *Part-of-speech tagging:* automatically identifies the morpho-syntactic categories (noun, verb, adjective) of words in the documents. The non-significant words can be filtered on the basis of their morpho-syntactic category. The part-of-speech tagging runs on the textual content i.e. on titles and abstracts of the patents.

    - *Part-of-speech selection:* to further reduce the vocabulary size, the user has the ability to select the word categories as identified by the assigned parts-of-speech and restrict the analysis to specific word categories (i.e. nouns, verbs, adjectives). Moreover, the user can select words that will not be involved in the subsequent steps of the analysis, and create synonyms.
- **Statistical analysis** consist in:
    - *Simple analysis.*
    - *Correspondence analysis* [3], [8].
    - *Cluster analysis* [1], [9].
    - *Bootstrap analysis.*

By using this methodology the system offers a stable and robust analysis of patent data by enabling the production of technology indicators through a fast and easy procedure. Moreover, the analysis of huge databases is possible in a realistic time of processing. There are no limits in the patent database used since the only requirement is to input the original data in txt format.

The technology used permits to provide high quality analysis in terms of patent processing, competitive capturing and capturing of technology watch. Below is given a brief description of system functioning and the subsequent processing steps:

1. Firstly, one has to import the data in the system and decide about the information that will be involved in the analysis. For example, one can select specific patents by defining filters in the different fields that describe a patent.
2. Then the process goes on by selecting a simple analysis or a more complex analysis. In the first case, the system analyses all the information as a whole and permits to produce simple graphs and tables. In the second case, a more complicated analysis is obtained based on a complex statistical methodology, which makes uses of linguistic pre-processing of the data [5], and then performs a correspondence, a cluster and a bootstrap analysis for the creation of group of patents that express similar technologies and the production of technology indicators. These procedures are completely automated in the system not requiring a deep knowledge of the subject and the only requirement is the definition of specific parameters.

In the sequence, the information is visualized through different ways consisting of graphs, tables and ready-made reports. The system enables the creation of homogeneous groups of patents that express similar technologies as well as specific technology indicators for each group. In addition it is possible for each group to exploit all relevant information to the patents belonging to this group i.e. information about assignees, inventors, countries, designated states, etc.

The system proved quire capable of giving answer to a set of different questions that usually arise about specific technologies and technology on goings through concrete and simple steps. The user will be really appealing by having a tool that is easy at use, does not require technical knowledge for using it and can be adapted to the users needs.

In analysing data, the system combines various kinds of analysis, and permits to capture technology trends, innovate technologies and hot technologies. In fact, though the developed statistical methodology consisting of linguistic pre-processing of the data, correspondence and cluster analysis for all the case studies we can obtain homogeneous groups of patents that express specific technological areas.

Furthermore, we can derive specific technology indicators for each cluster. These are of great importance since permit to quantify all the information related to the specific technologies. Therefore, we can obtain indicators about the patenting activity of specific actors, inventors and countries. We are also able to derive technology indicators explaining the promising technologies for a wide range of technology activities, the innovative technologies, as well as trace the most important actors in these areas of technology. It is also possible to derive technology indicators over time. More specifically we can check for the specific technologies their evolution over time, and also be able to see for specific companies their activity in time. The same information was easily obtained for specific countries or continents.

Although the production of technology indicators is a rather complicated procedure, by using the specific system we are able to produce several technology indicators in a fast way through discrete steps.

In terms of the visualization of the results it is realized how important is to use a system that does not only offers complex statistical procedures and analysis but also enable in an easy way to depict the information in a way easily understood. The specific system contains many options in terms of the offered visualization techniques that allow each time to obtain the most appropriate output. These outputs were graphs, tables or ready-made reports.

4 Evaluation

Within the prototyping phase, we planned a working demonstration system. The pilot applications provided the infrastructure in which the system was tested in a technical and functional way. The demonstrations trials that took place aimed at two things: firstly they enabled to test the value of the use of the system at a variety of statistical themes related to the needs of the users. Secondly it was tested the usefulness of the system across a range of different users.

For this purpose the system was presented to the different user group meetings and in several events. The specific demonstration aimed at clarifying to the users, which are the functionalities of the system, what kinds of analyses are available as well as to present the visualization techniques for the formatting of the produced results. Therefore, this presentation permitted to become familiar with the system and understand the way it works. The case studies were based on real patent data and during the demonstrations in the user group meetings the users had the opportunity to give answers to specific questions to technology problems in the technology domain they are specialized in.

As a conclusion taking into account the feedback from the users we mention that the adopted technology permits to obtain high quality results in terms of patent processing, competitive analyses and capturing of technology watch. The key benefits that the system offers to the end-users are summarized below:

- **Save time.** Generally, patent data statistical analysis requires considerable time. The system allows users to conduct analyses quickly and efficiently as it provides them with a variety of tools and functionalities (filters, charts, tables, ready-made reports, etc.).
- **Ease-of-use.** It contains user interface enhancements and features that improve and support users and make analyses easy and flexible.
- **Data correction.** The system allows users to explore and even correct every filed in patent data. Moreover, users can define new fields, keep annotations on them, etc., so that they can add value to the downloaded data.
- **Wizards.** Wizards leads user step by step to analyse patent data. Even a novice user will be able to analyse information and create sophisticated charts only after a few clicks.
- **Visualization tools.** It offers a great variety of tools in order to visualize the analysis results (e.g. ready-made reports in various formats). So, the user with only a few clicks can quickly get the overall picture of his/her data.
- **Export/import capability.** It offers the user the ability to import patent data from a variety of International Patent Databases and export the results of his analysis in various and well-known file formats.
- **Automatic/manual report generation.** A powerful tool for producing reports gives the user the ability to summarize the patent data analysis procedure and its results in reports available for printing.
- **Manipulation of large database.** The system allows users to explore and efficiently analyse even large databases of patent data making analysis an easy, flexible and quick procedure.

5. Conclusions

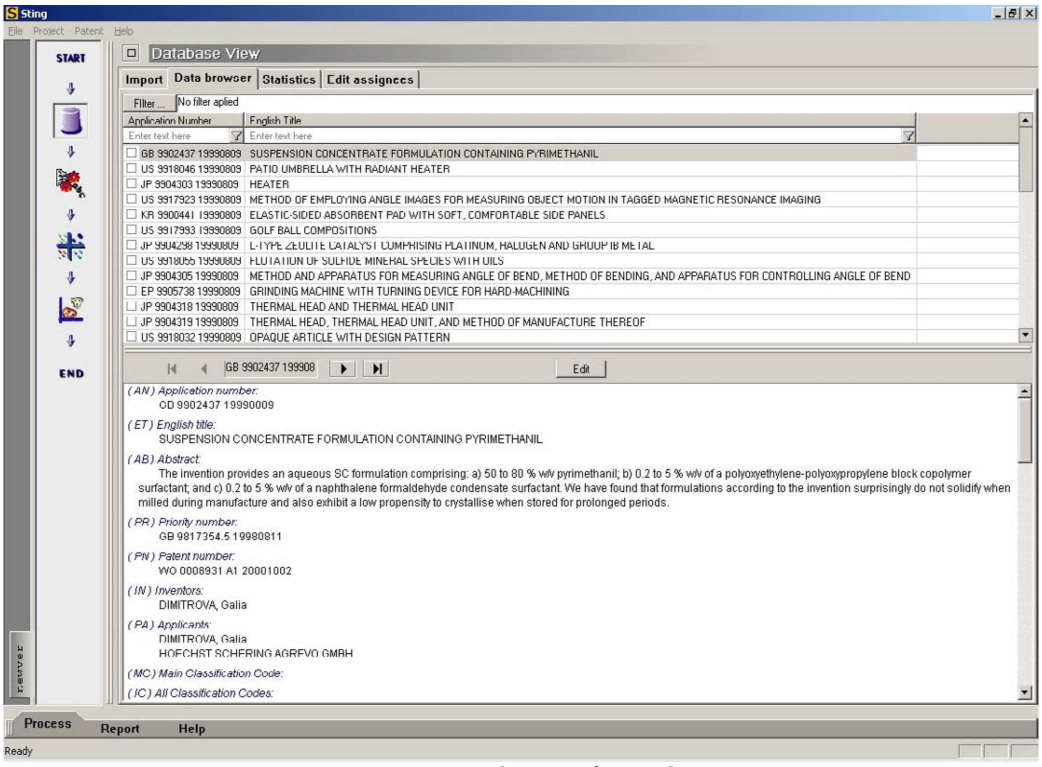
The presented system accepts data from any patent database and proceeds with a set of complicated analysis without requiring any special qualifications from the end-user. The user is able to select the kind of analysis he is interested and through successive steps perform a complete analysis and obtain robust results. Furthermore, the user is able to totally intervene in the produced results in order to visualize the information in his own prospect of view and finally he is able to create his own ready-made reports. Another feature that makes this tool competitive is the background technologies used, which combine both innovate IT technologies as well as complex statistical methodologies.

Finally, the system maximizes the user's productivity and satisfaction by providing him not only with the correct guidance but also with tools to expand the system in a suitable patent data analysis environment. This means that the system enables the user to extract only necessary information and exploit it in an informative way in order to draw useful conclusions.

References

1. Alderferer, M.S., Blashfield, R.K. Cluster Analysis, Beverly Hills, CA., Sage Publications, Inc., 1986.
2. Beaugrande, R., Dressler, W. Introduction to Text Linguistics, London Longman, 1981.
3. Benzecri, J.P. Correspondence Analysis Handbook, New York: Marcel Dekker, 1992.
4. Chappelier, J., Peristera, V., Rajman, M., Seydoux, F. Evaluation of Statistical and Technological Innovation Using Statistical Analysis of Patents, JADT 2002.
5. Ciravegna, F., Lavelli, A., Pianesi, F. Linguistic Processing of Texts Using Geppetto, Technical Report 9602-06, IRST, Povo TN, Italy, 1996.
6. Comanor, W.S., Scherer, F.M. Patent Statistics as a Measure of Technical Change. Journal of Political Economy, pp. 392-398, 1969.
7. EPO- European Patent Office. Available at:http://www.european-patent-office.org/index.htm.
8. Hill, M.O. Correspondence Analysis: a Neglected Multivariate Method. Journal of Applied Statistics, Vol. 23, No. 3, pp. 340-354, 1974.
9. Lewis, S. Cluster Analysis as a Technique to Guide Interface Design, International Journal of Man-Machine Studies, 35, pp. 251-265, 1991.
10. Narin, F. Patents as Indicators for the Evaluation of IndustrialResearch Output. Scientometrics, 34, 3, pp. 489-496, 1995.
11. Schmoch, U., Bierhals, R., Rangnow, R. Impact of International Patent Applications on Patent Indicators. JOINT NESTI/TIP/GSS WORKSHOP, Room Document No. 1, 1998.

Appendix



No filter aplied

Application Number	English Title
Enter text here	Enter text here
GB 9902437 19990809	SUSPENSION CONCENTRATE FORMULATION CONTAINING PYRIMETHANIL
US 9918046 19990809	PATIO UMBRELLA WITH RADIANT HEATER
JP 9904303 19990809	HEATER
US 9917923 19990809	METHOD OF EMPLOYING ANGLE IMAGES FOR MEASURING OBJECT MOTION IN TAGGED MAGNETIC RESONANCE IMAGING
KR 9900441 19990809	ELASTIC-SIDED ABSORBENT PAD WITH SOFT, COMFORTABLE SIDE PANELS
US 9917993 19990809	GOLF BALL COMPOSITIONS
JP 9904298 19990809	L-TYPE ZEOLITE CATALYST COMPRISING PLATINUM, HALOGEN AND GROUP IB METAL
US 9918055 19990809	FLOTATION OF SULFIDE MINERAL SPECIES WITH OILS
JP 9904305 19990809	METHOD AND APPARATUS FOR MEASURING ANGLE OF BEND, METHOD OF BENDING, AND APPARATUS FOR CONTROLLING ANGLE OF BEND
EP 9905738 19990809	GRINDING MACHINE WITH TURNING DEVICE FOR HARD-MACHINING
JP 9904318 19990809	THERMAL HEAD AND THERMAL HEAD UNIT
JP 9904319 19990809	THERMAL HEAD, THERMAL HEAD UNIT, AND METHOD OF MANUFACTURE THEREOF
US 9918032 19990809	OPAQUE ARTICLE WITH DESIGN PATTERN

GB 9902437 199908

(AN) Application number:
GB 9902437 19990809

(ET) English title:
SUSPENSION CONCENTRATE FORMULATION CONTAINING PYRIMETHANIL

(AB) Abstract:
The invention provides an aqueous SC formulation comprising: a) 50 to 80 % w/v pyrimethanil; b) 0.2 to 5 % w/v of a polyoxyethylene-polyoxypropylene block copolymer surfactant; and c) 0.2 to 5 % w/v of a naphthalene formaldehyde condensate surfactant. We have found that formulations according to the invention surprisingly do not solidify when milled during manufacture and also exhibit a low propensity to crystallise when stored for prolonged periods.

(PR) Priority number:
GB 9817354.5 19980811

(PN) Patent number:
WO 0008931 A1 20001002

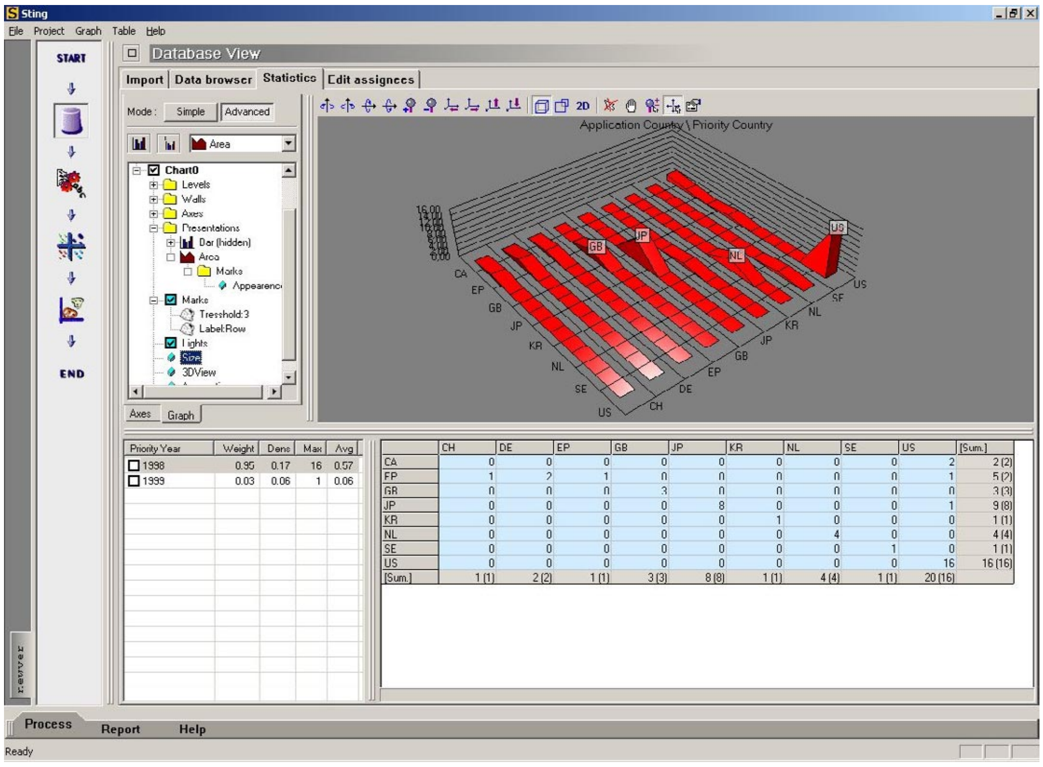
(IN) Inventors:
DIMITROVA, Galia

(PA) Applicants:
DIMITROVA, Galia
HOECHST SCHERING AGREVO GMBH

(MC) Main Classification Code:

(IC) All Classification Codes:

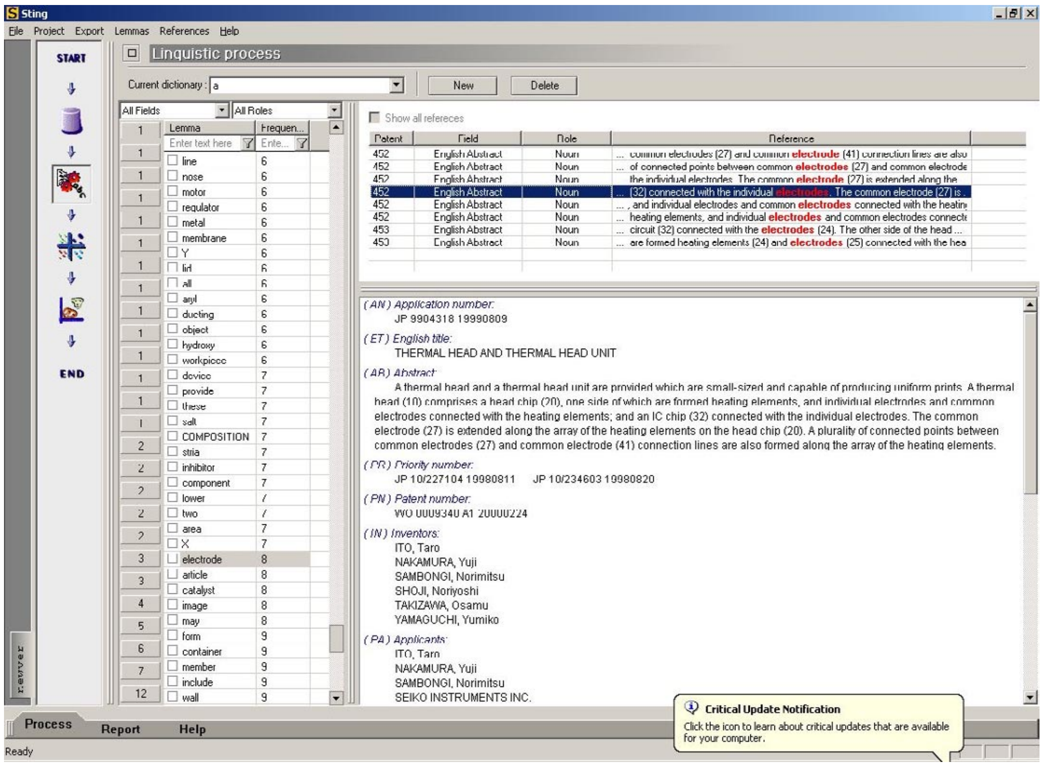
Figure 3: Visualization of input data



Priority Year	Weight	Dens	Max	Avg
1998	0.95	0.17	16	0.57
1999	0.03	0.06	1	0.06

	CH	DE	EP	GB	JP	KR	NL	SE	US	[Sum.]
CA	0	0	0	0	0	0	0	0	2	2 (2)
EP	1	2	1	0	0	0	0	0	1	5 (2)
GB	0	0	0	3	0	0	0	0	0	3 (3)
JP	0	0	0	0	8	0	0	0	1	9 (8)
KR	0	0	0	0	0	1	0	0	0	1 (1)
NL	0	0	0	0	0	0	4	0	0	4 (4)
SE	0	0	0	0	0	0	0	1	0	1 (1)
US	0	0	0	0	0	0	0	0	16	16 (16)
[Sum.]	1 (1)	2 (2)	1 (1)	3 (3)	8 (8)	1 (1)	4 (4)	1 (1)	20 (16)	

Figure 4: Simple statistics and graphical representation



Lemma	Frequen...
line	6
nose	6
motor	6
regulator	6
metal	6
membrane	6
Y	6
lid	6
all	6
aryl	6
ducting	6
object	6
hydroxy	6
workpiece	6
device	7
provide	7
these	7
salt	7
COMPOSITION	7
stria	7
inhibitor	7
component	7
lower	7
two	7
area	7
X	7
electrode	8
article	8
catalyst	8
image	8
may	8
form	9
container	9
member	9
include	9
wall	9

Patent	Field	Role	Reference
452	English Abstract	Noun	... common electrodes (27) and common **electrode** (41) connection lines are also
452	English Abstract	Noun	... of connected points between common **electrodes** (27) and common electrode
452	English Abstract	Noun	the individual electrodes. The common **electrode** (27) is extended along the
452	English Abstract	Noun	... (32) connected with the individual **electrodes**. The common electrode (27) is .
452	English Abstract	Noun	... , and individual electrodes and common **electrodes** connected with the heatin
452	English Abstract	Noun	... heating elements, and individual **electrodes** and common electrodes connect
453	English Abstract	Noun	... circuit (32) connected with the **electrodes** (24). The other side of the head ...
453	English Abstract	Noun	... are formed heating elements (24) and **electrodes** (25) connected with the hea

(AN) Application number:
JP 9904318 19990809

(ET) English title:
THERMAL HEAD AND THERMAL HEAD UNIT

(AB) Abstract:
A thermal head and a thermal head unit are provided which are small-sized and capable of producing uniform prints. A thermal head (10) comprises a head chip (20), one side of which are formed heating elements, and individual electrodes and common electrodes connected with the heating elements; and an IC chip (32) connected with the individual electrodes. The common electrode (27) is extended along the array of the heating elements on the head chip (20). A plurality of connected points between common electrodes (27) and common electrode (41) connection lines are also formed along the array of the heating elements.

(PR) Priority number:
JP 10/227104 19980811 JP 10/234603 19980820

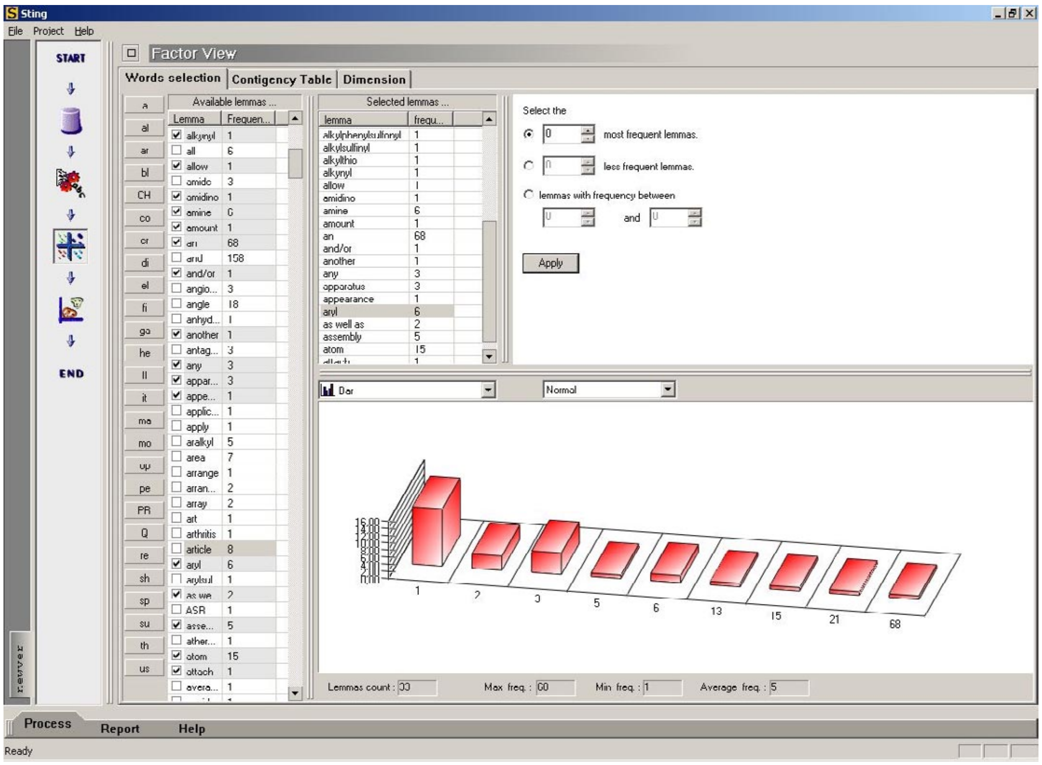
(PN) Patent number:
WO 0009340 A1 20000224

(IN) Inventors:
ITO, Taro
NAKAMURA, Yuji
SAMBONGI, Norimitsu
SHOJI, Noriyoshi
TAKIZAWA, Osamu
YAMAGUCHI, Yumiko

(PA) Applicants:
ITO, Taro
NAKAMURA, Yuji
SAMBONGI, Norimitsu
SEIKO INSTRUMENTS INC.

Critical Update Notification
Click the icon to learn about critical updates that are available for your computer.

Figure 5: Linguistic processing



Words selection | Contigency Table | Dimension

a · al · ar · bl · CH · co · cr · di · el · fi · go · he · ll · it · ma · mo · up · pe · PR · Q · re · sh · sp · su · th · us

Available lemmas ...

	Lemma	Frequen...
✓	alkynyl	1
☐	all	6
✓	allow	1
☐	amido	3
✓	amidino	1
✓	amine	6
✓	amount	1
✓	an	68
☐	and	158
✓	and/or	1
☐	angio...	3
☐	angle	18
☐	anhyd...	1
✓	another	1
☐	antag...	3
✓	any	3
✓	appar...	3
✓	appe...	1
☐	applic...	1
☐	apply	1
☐	aralkyl	5
☐	area	7
☐	arrange	1
☐	arran...	2
☐	array	2
☐	art	1
☐	arthritis	1
☐	article	8
✓	aryl	6
☐	arylsul	1
✓	as we	2
☐	ASR	1
✓	asse...	5
☐	ather...	1
✓	atom	15
✓	attach	1
☐	avera...	1

Selected lemmas ...

lemma	frequ...
alkylphenylsulfonyl	1
alkylsulfinyl	1
alkylthio	1
alkynyl	1
allow	1
amidino	1
amine	6
amount	1
an	68
and/or	1
another	1
any	3
apparatus	3
appearance	1
aryl	6
as well as	2
assembly	5
atom	15

Select the

[✓ 0 most frequent lemmas.]

0 less frequent lemmas.

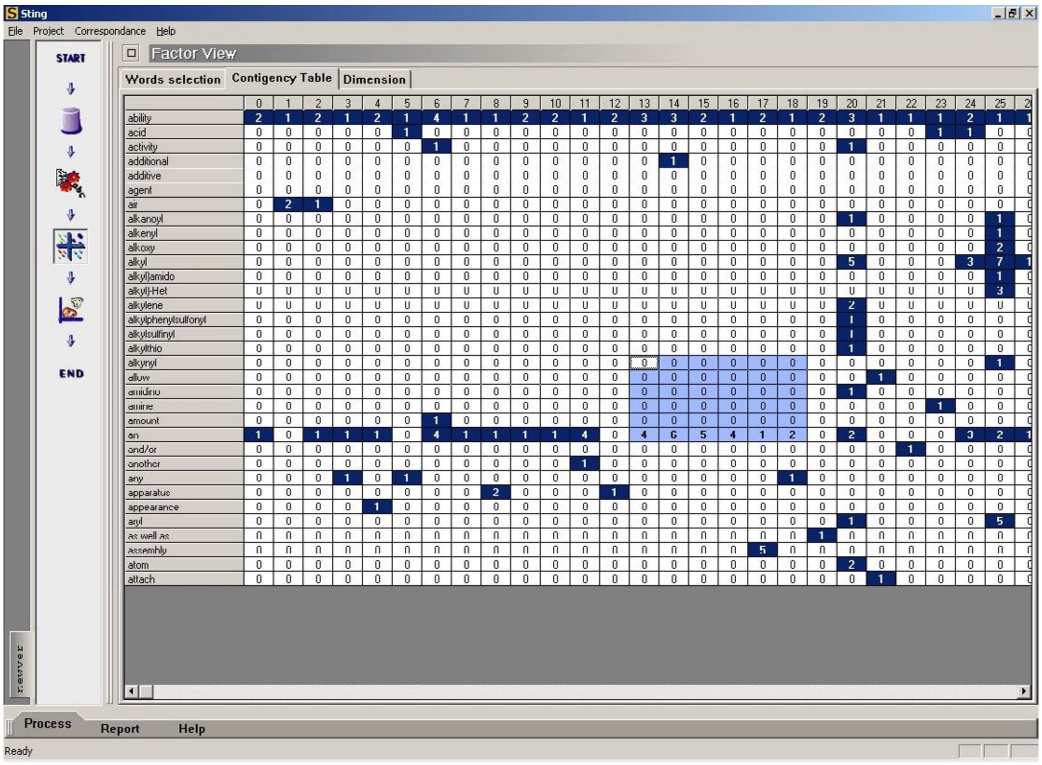
lemmas with frequency between 0 and 0

Apply

Bar · Normal

Lemmas count : 33 · Max freq. : 68 · Min freq. : 1 · Average freq. : 5

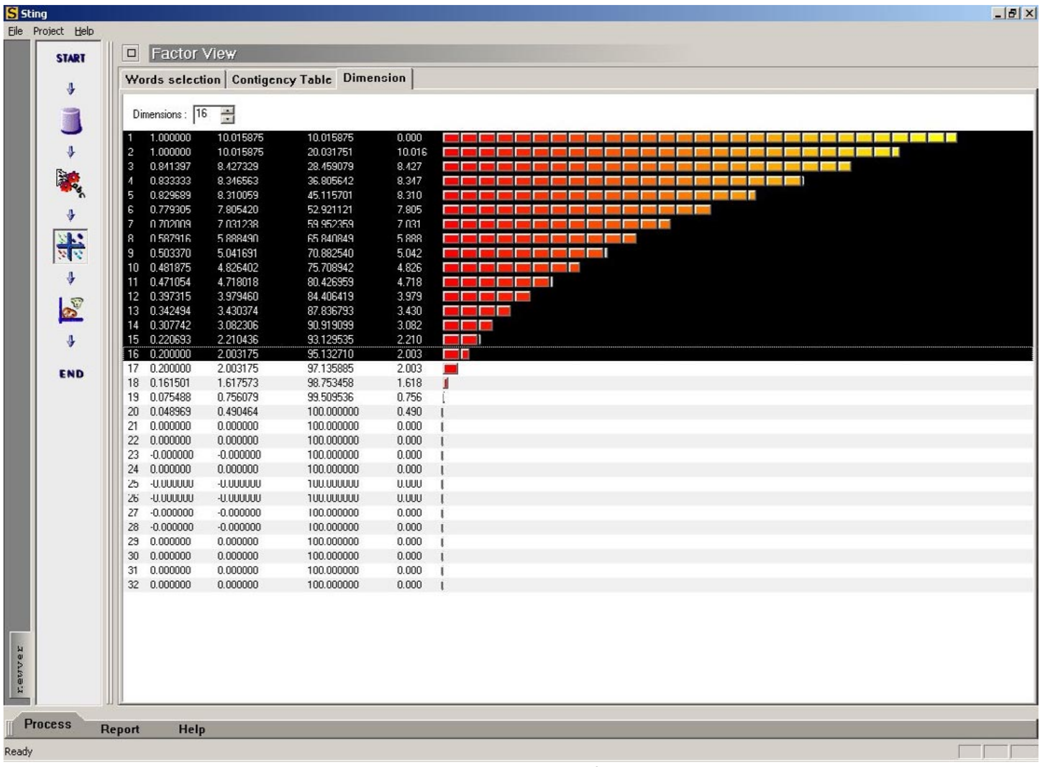
Figure 6: Selection of lemmas and graphical representation



Words selection | Contigency Table | Dimension

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	2
ability	2	1	2	1	2	1	4	1	1	2	2	1	2	3	3	2	1	2	1	2	3	1	1	1	2	1	1
acid	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0
activity	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
additional	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
additive	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
agent	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
air	0	2	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
alkanoyl	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1	0
alkenyl	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
alkoxy	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0
alkyl	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	5	0	0	0	3	7	1
alkyl)amido	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
alkyl)-Het	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U	3	U
alkylene	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U	2	U	U	U	U	U	U
alkylphenylsulfonyl	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
alkylsulfinyl	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
alkylthio	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
alkynyl	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
allow	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
amidino	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
amine	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
amount	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
an	1	0	1	1	1	0	4	1	1	1	1	4	0	4	6	5	4	1	2	0	2	0	0	0	3	2	1
and/or	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
another	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
any	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
apparatus	0	0	0	0	0	0	0	0	2	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
appearance	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
aryl	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	5	0
as well as	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
assembly	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	5	0	0	0	0	0	0	0	0	0
atom	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0
attach	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0

Figure 7: Contingency matrix involved in the correspondence and cluster analysis.



Dimensions : 16

1	1.000000	10.015875	10.015875	0.000
2	1.000000	10.015875	20.031751	10.016
3	0.841397	8.427329	28.459079	8.427
4	0.833333	8.346563	36.805642	8.347
5	0.829689	8.310059	45.115701	8.310
6	0.779305	7.805420	52.921121	7.805
7	0.702009	7.031238	59.952359	7.031
8	0.587916	5.888490	65.840849	5.888
9	0.503370	5.041691	70.882540	5.042
10	0.481875	4.826402	75.708942	4.826
11	0.471054	4.718018	80.426959	4.718
12	0.397315	3.979460	84.406419	3.979
13	0.342494	3.430374	87.836793	3.430
14	0.307742	3.082306	90.919099	3.082
15	0.220693	2.210436	93.129535	2.210
16	0.200000	2.003175	95.132710	2.003
17	0.200000	2.003175	97.135885	2.003
18	0.161501	1.617573	98.753458	1.618
19	0.075488	0.756079	99.509536	0.756
20	0.048969	0.490464	100.000000	0.490
21	0.000000	0.000000	100.000000	0.000
22	0.000000	0.000000	100.000000	0.000
23	-0.000000	-0.000000	100.000000	0.000
24	0.000000	0.000000	100.000000	0.000
25	-0.000000	-0.000000	100.000000	0.000
26	-0.000000	-0.000000	100.000000	0.000
27	-0.000000	-0.000000	100.000000	0.000
28	-0.000000	-0.000000	100.000000	0.000
29	0.000000	0.000000	100.000000	0.000
30	0.000000	0.000000	100.000000	0.000
31	0.000000	0.000000	100.000000	0.000
32	0.000000	0.000000	100.000000	0.000

Figure 8: Histogram of indices

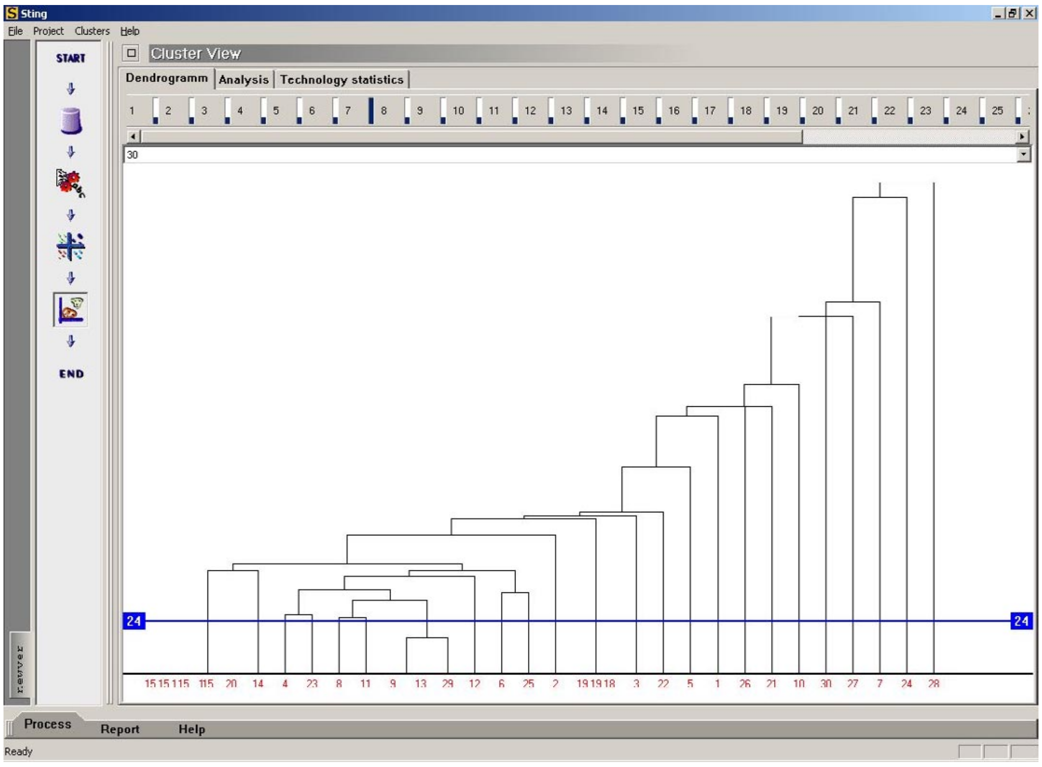


Figure 9: Dendrogram

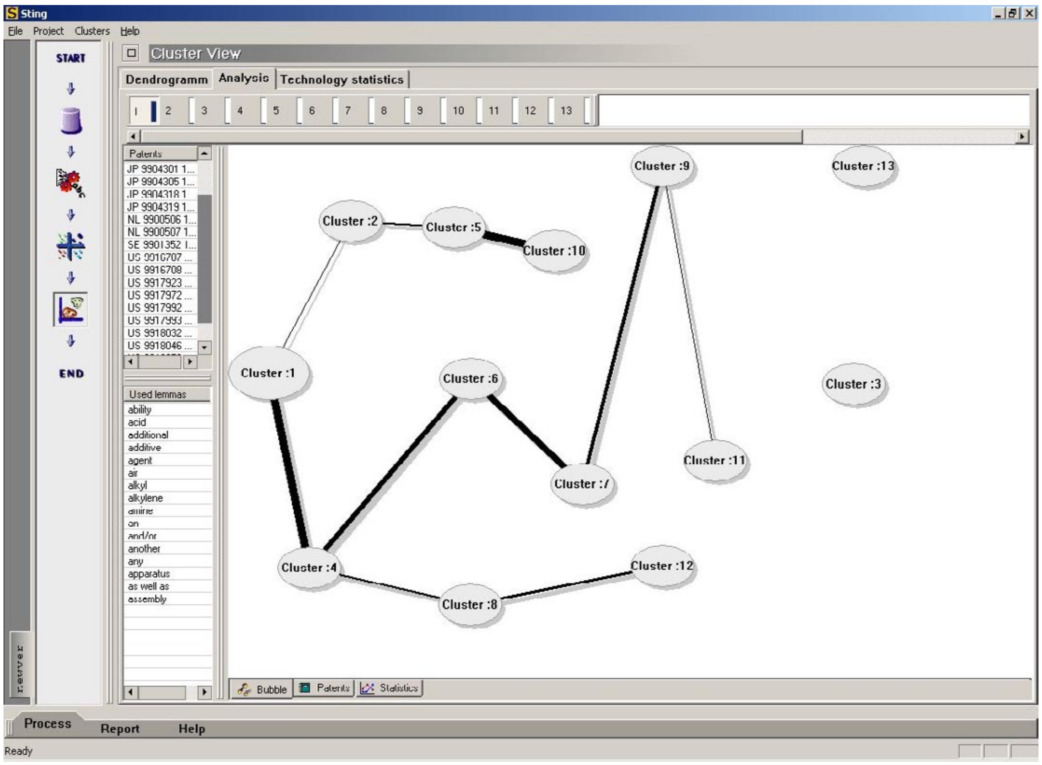
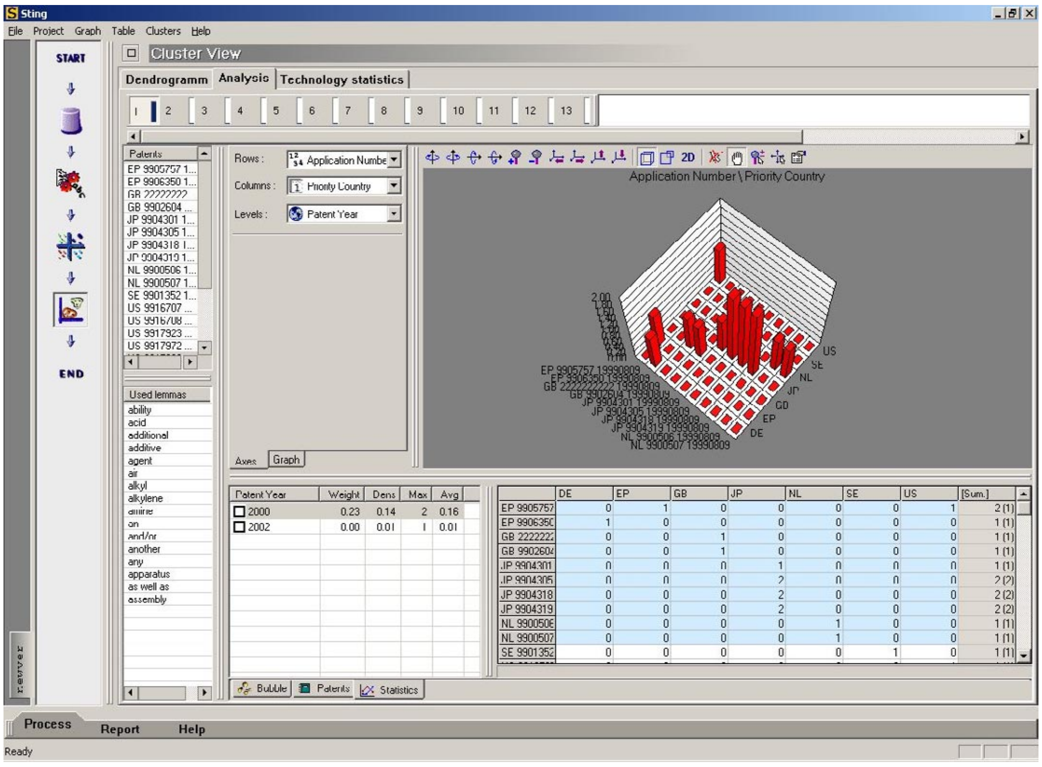


Figure 10: Cluster map



Patent Year	Weight	Dens	Max	Avg
2000	0.23	0.14	2	0.16
2002	0.00	0.01	1	0.01

	DE	EP	GB	JP	NL	SE	US	[Sum.]
EP 9905757	0	1	0	0	0	0	1	2 (1)
EP 9906350	1	0	0	0	0	0	0	1 (1)
GB 2222222	0	0	1	0	0	0	0	1 (1)
GB 9902604	0	0	1	0	0	0	0	1 (1)
JP 9904301	0	0	0	1	0	0	0	1 (1)
JP 9904305	0	0	0	2	0	0	0	2 (2)
JP 9904318	0	0	0	2	0	0	0	2 (2)
JP 9904319	0	0	0	2	0	0	0	2 (2)
NL 9900506	0	0	0	0	1	0	0	1 (1)
NL 9900507	0	0	0	0	1	0	0	1 (1)
SE 9901352	0	0	0	0	0	1	0	1 (1)

Figure 11: Visualization of the results per cluster