YALE: Rapid Prototyping for Complex Data Mining Tasks

Ingo Mierswa
Artificial Intelligence Unit
Department of Computer Science
University of Dortmund
ingo.mierswa@uni-dortmund.de

Michael Wurst
Artificial Intelligence Unit
Department of Computer Science
University of Dortmund
wurst@ls8.cs.uni-dortmund.de

Ralf Klinkenberg
Artificial Intelligence Unit
Department of Computer Science
University of Dortmund
ralf.klinkenberg@uni-dortmund.de

Martin Scholz
Artificial Intelligence Unit
Department of Computer Science
University of Dortmund
scholz@ls8.cs.uni-dortmund.de

Timm Euler
Artificial Intelligence Unit
Department of Computer Science
University of Dortmund
timm.euler@uni-dortmund.de

ABSTRACT

KDD is a complex and demanding task. While a large number of methods has been established for numerous problems, many challenges remain to be solved. New tasks emerge requiring the development of new methods or processing schemes. Like in software development, the development of such solutions demands for careful analysis, specification, implementation, and testing. Rapid prototyping is an approach which allows crucial design decisions as early as possible. A rapid prototyping system should support maximal re-use and innovative combinations of existing methods, as well as simple and quick integration of new ones.

This paper describes YALE, a free open-source environment for KDD and machine learning. YALE provides a rich variety of methods which allows rapid prototyping for new applications and makes costly re-implementations unnecessary. Additionally, YALE offers extensive functionality for process evaluation and optimization which is a crucial property for any KDD rapid prototyping tool. Following the paradigm of visual programming eases the design of processing schemes. While the graphical user interface supports interactive design, the underlying XML representation enables automated applications after the prototyping phase.

After a discussion of the key concepts of YALE, we illustrate the advantages of rapid prototyping for KDD on case studies ranging from data pre-processing to result visualization. These case studies cover tasks like feature engineering, text mining, data stream mining and tracking drifting concepts, ensemble methods and distributed data mining. This variety of applications is also reflected in a broad user base, we counted more than 40,000 downloads during the last twelve months.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
KDD'06, August 20–23, 2006, Philadelphia, Pennsylvania, USA.
Copyright 2006 ACM 1-59593-339-5/06/0008 ...$5.00.

Track: Industrial Track

Categories and Subject Descriptors: I.5.2 [Computing Methodologies]: Pattern Recognition

General Terms: Design, Experimentation

Keywords: KDD system, rapid prototyping, multimedia mining, audio and text mining, data stream mining, data pre-processing, result visualization, distributed data mining, feature construction

1. INTRODUCTION

It is well known that knowledge discovery (KD) is a highly complex process. Like software development, it requires careful analysis, specification, implementation and testing. Prototyping plays an important role in this process. On the one hand, prototyping helps to identify adequate methods and optimal parameters. This enables developers to make crucial design decisions as early as possible in the knowledge discovery process. Costly redesign at later stages can be avoided. On the other hand, prototyping helps to control several risks. Most importantly, the performance of the envisioned system can be estimated beforehand. This gives the customer an impression of the final result and its limitations. It also helps to clarify misunderstandings concerning the envisioned outcome. Another important aspect is to estimate computation time and cost of the final system. Especially for applications with tight constraints on these resources (e.g. real time systems), such an estimation is essential in order to decide to which extent knowledge discovery can be applied.

A prototyping framework for knowledge discovery must meet several requirements. First, it should be very flexible with respect to methods for preprocessing and data analysis. This implies that it must offer a very broad range of differ-

ent methods and that incorporating new methods is easy. Also, it should be able to process different kinds of input data, as time series or text data, without additional effort for the user. Second, a rapid prototyping tool for knowledge discovery must provide extensive functionality for evaluation and optimization. Finally, prototyping tools must be very easy to use. In particular, they should not require the user to learn a complex formalism. Many prototyping tools therefore employ the paradigm of visual programming.

The YALE system[1] was developed to meet the above requirements. YALE is an environment for machine learning experiments and data mining supporting the paradigm of rapid prototyping. Experiments can be made up of a large number of arbitrarily nestable operators and their setup is described by XML files which can easily be created with a graphical user interface. Applications of YALE cover both research and real-world data mining tasks.

2. BASIC CONCEPTS

Real-world knowledge discovery processes typically consist of one or more data pre-processing, machine learning, evaluation, and visualization steps. Hence, a data mining platform should allow complex experiment designs, transparent data handling, comfortable parameter handling and optimization, flexible rearrangements, and extendibility. Finally, a rapid prototyping environment for knowledge discovery must ensure that even complex experiments can easily be designed. In this section we will discuss some of the basic concepts of YALE which eases the design of new methods.

2.1 Modeling knowledge discovery processes as operator trees

Knowledge discovery (KD) processes are often viewed as sequential invocations of single methods. For example, after loading the data, one might apply a preprocessing step followed by an invocation of a classification learning method. The result in this case is the learned model, which can be applied on new and unseen data. A possible abstraction of these single methods is the concept of *operators*. Each operator receives its input, performs a defined action and delivers some output. Hence, the sequential method invocations correspond to an operator chain. Although this chain model is sufficient for many basic knowledge discovery tasks, flat chains are often insufficient to model complex KD processes.

A common approach for more complex experiment designs is designing the operator combinations as directed graph. Each vertex of the graph corresponds to a single operator. If two operators are connected, the output of the first operator should be used as input of the second. On the one hand, designing knowledge discovery processes with help of directed graphs is very powerful. On the other hand, it has a main drawback: due to missing restrictions and necessary topological sorting the design of experiments is often not intuitive and automatic validations are harder to perform.

YALE offers a compromise between the simplicity of operator chains and the power of directed graphs by modeling KD processes as *operator trees*. Similar to programming languages, using operator trees allows concepts like loops, conditions, or other meta application schemes. The leafs in the operator tree correspond to simple steps in the modeled process like learning a prediction model or applying a preprocessing filter. Inner nodes of the tree correspond to more complex or abstract steps in the process. This is often necessary if the children should be applied several times like, for example, in loops. In general, inner operator nodes define the data flow through their children. The root of the tree corresponds to the whole experiment.

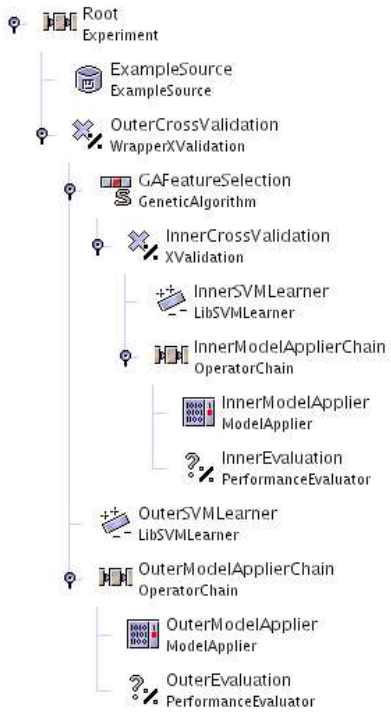

Figure 1: The operator tree for a complex data mining experiment. The upper label denotes the operator instance name, the lower label the operator class name.

Figure 1 shows a nested KD process for feature selection using a genetic algorithm with an inner cross-validation for evaluating candidate feature sets and an outer cross-validation for evaluating the genetic algorithm as a feature selector. This setup is modeled as an operator tree. The data flow is the same as for depth first search (DFS) with only one exception. If an inner operator node performs a loop, the input and output objects might be passed more than once to the children before the final output is generated and given to the next sibling of the parent.

Operators define their expected inputs and delivered outputs as well as their obligatory and optional parameters, which enables YALE to automatically check the nesting of operators, the types of the objects passed between the operators, and the mandatory parameters. This eases the design of complex data mining experiments. Following the concept of visual programming, users can easily create operator trees with the help of a graphical user interface. Break points before or after operators can be used to check intermediate results and the data flow. This eases both debugging and interpreting the results.

YALE internally uses XML to describe the operator trees which is easily readable by humans and machines. The XML description of an experiment consists of nested tag elements

[1] `http://yale.sf.net`

for all operators. Each operator element defines its instance name and operator class via XML attributes and the parameters via inner tags. This XML experiment configuration define an interchange format for data mining experiments and also ensures the reproducibility of experiments. Since default parameters do not need to be specified, the XML description is a very concise representation of the operator tree structure.

The XML experiment configurations can be used as a *scripting language* for KD experiments. Each operator corresponds to a single step or function invocation like preprocessing or model learning. In consequence, the data mining step of other programs can easily be configured by alterations of the basic XML scripts. This property was for example used by the Network Media Organizer NEMOZ[2].

2.2 Multi-layered data view concept

YALE's most important characteristic is the ability to nest operator chains and build complex operator trees. In order to support this functionality the YALE data core acts like a data base management system and provides a *multi-layered data view* on a common data table which underlies all views. The views on this table do not contain the data itself but only references on rows or columns of this table. In case of a data set, views on the rows of the table correspond to subsets of the data, and views on the columns correspond to the selected features used to represent the examples.

All views are maintained by a stack of views. For example, the first view can select a subset of examples and the second view can select a subset of features. The result is a single view which reflects both views. The number of layered views is virtually not limited. Other supported views include conditioned data set filters allowing only examples fulfilling a given condition, missing value filters, weighting views applying a weight on examples or features, or views creating new attributes on the fly.

This multi-layered view concept is a very efficient way to store different views on the same data table. This is especially important for automatic data preprocessing tasks like feature generation or selection. In order not to unnecessarily copy the data set or subsets of it, YALE manages views on this table, so that only references to the relevant parts of the table need to be copied or passed between operators. For example, the population of an evolutionary operator may consist of several data views - instead of several duplicates of parts of the data set.

2.3 Data handling

No matter whether a data set is stored in memory, in a file, or in a database, YALE internally uses special types of data tables to access the data via an uniform external interface. The data handling is hence *transparent* to the operators. YALE achieves this transparent data handling by supporting several types of data sources and hiding internal data transformations and partitioning from the user or other operators. Therefore, operators do not have to cope with the actual data format or different data views.

In order to guide transformations of the feature space or the automatic search for the best preprocessing, the user can define additional *meta data*. Meta data include the type of attributes or their unit according to the international system of units (SI). This information is for example used by feature

[2] `http://nemoz.sf.net`

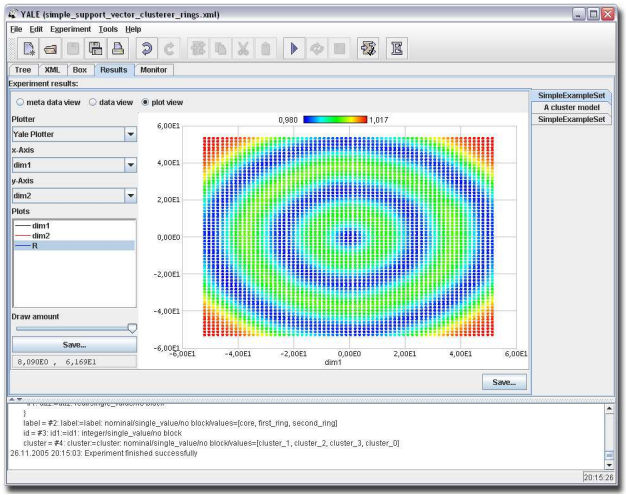

Figure 2: A 2D plot of the radius function determined with a Support Vector Clustering algorithm. This operator is part of the Clustering plugin.

generation algorithms in order to prevent the aggregation of features which do not have the same unit of measurement. The definition of meta information is optional, YALE tries to guess the correct data types automatically.

2.4 Built-in operators and extensions

A rapid prototyping environment for knowledge discovery should already support a huge number of operations and state of the art methods. For example, Figure 2 shows the visualization of the result of a support vector clusterer. Together with the available plugins, YALE provides more than 400 operators. This set of operators include a wide range of *in- and output operators*, (Arff, C4.5, csv, or sparse formats, databases like Oracle, mySQL, or Postgres, and other file formats like dBase, text and audio files etc.), *machine learning* operators (support vector machines, decision tree and rule learners, lazy learners, bayesian learners, logistic regression, gaussian processes, meta learning techniques, association rule mining algorithms and clustering schemes etc.), *data preprocessing* operators (discretization, example and feature filtering, normalization, sampling, dimensionality reduction, and additional and infinite value replenishment filters etc.), *feature space transformation* operators (forward selection, backward elimination, genetic algorithms, weight guided, feature weighting and relevance calculation, feature construction and extraction etc.), *evaluation* operators and *meta optimization* operators (parameter optimization or experiment loops etc.). Of course, a wide range of *visualization* tools like online plots of your data or experiment results are also provided.

Although YALE provides this broad variety of operators for different data mining tasks, it might sometimes be necessary to implement a missing functionality. In order to implement a new operator, the developer simply needs to define the expected inputs, the delivered outputs, the mandatory and optional parameters, and the core functionality of the operator. YALE provides an exhaustive application programming interface (API) with clean interfaces in order to allow quick access to the main methods. The operator description

in XML allows YALE to automatically create corresponding GUI elements. An easy-to-use plugin mechanism is provided in order to add functionality in a modular way which allows contributions by the YALE community that do not require changes in the YALE core. Several plugins are available in the download section of our web site.

Beside the graphical and the command line interface, YALE can also be directly used from other programs. Clear programming interfaces define an easy way of applying single operators, operator chains, or complete operator trees on input data. Both the command line version and a Java API allow the invocation of YALE from programs without using the GUI. Since YALE is entirely written in Java, it runs on any major platform and operating system.

3. CASE STUDIES

In the next section, we present several case studies which exemplify the basic design concepts of YALE and show their relation to a broad variety of knowledge discovery tasks.

3.1 Case study 1: Ensemble learning on large databases

For many data mining applications, the predictive performance of a model can be improved drastically by considering several base learners in parallel. The use of ensemble methods is especially appealing if the available amount of training data is huge, which is true for most real-world knowledge discovery tasks. As a consequence of the super-linear runtime of most learning algorithms, learning from all the data is often too costly. A common work-around is to learn an ensemble from a single subsample. This often yields suboptimal results, because it does not take full advantage of the available training data.

For large-scale data mining, YALE offers a direct JDBC interface to relational databases. This eases the induction of base models from subsamples of tractable size. Moreover, due to the nestable operators, the induction of individual base models, each based on a separate database subsample, is possible with negligible efforts. This strategy prevents overfitting and circumvents the problems of scalability without waisting precious information during model building.

A strong point of YALE in the context of ensemble methods is the fact that all predictions and associated confidences are handled the same way as regular features are, but without losing their predictive semantics during cross-validations and similar operations. Hence, if required, predictions and associated confidence values may be used as regular base features for subsequent learning steps. This facilitates stacking, the optimization of base model weights for boosting, and it eases the integration of novel ensemble methods as YALE operators for prototypical evaluation.

During the last years a variety of ensemble algorithms has been evaluated using the YALE framework. The Bayesian Boosting operator can optionally start from a given previously constructed ensemble, e.g. the result of a previous operator or a model read from file, and use it to weight the data at hand according to its current predictive performance. It then augments the ensemble by adding additional base models as required for the task. This way an ensemble of heterogeneous base learners can be constructed.

Apart from classification, the task of sequential subgroup discovery for given prior knowledge has recently been addressed successfully [18].

3.2 Case study 2: Text mining and tracking drifting concepts

Text mining covers a wide range of topics both in terms of methods and applications. YALE and its plugins offer tools to tackle many of them. This includes e.g. information filtering of news texts for the automated generation of personalized news from multiple sources, the automated sorting of documents or web pages into pre-defined categories, or e-mail routing, i.e. the automatic forwarding of e-mail messages to the most appropriate person or department in a company or organization.

Text mining usually starts with data pre-processing. Textual data often is unstructured, e.g. free text in natural language, or semi-structured, like e.g. in HTML documents, i.e. pages in the World Wide Web, or in XML documents. Many machine learning methods are designed for example sets with examples in an attribute value vector representation. Hence the text data needs to be pre-processed, e.g. by transforming the free text in documents into document vectors in the vector space model [17]. In the vector space model (*bag of words model*), the order of the words in a document is ignored. Each word occurring in a document is considered as an attribute, and the frequency of this word in a particular document is taken as the value of this attribute for this document. Very frequent and hence uninformative words (stop words) like articles, prepositions, etc. are often removed. Afterwards, these term frequency (TF) values are often weighted to down-weight frequent words, e.g. by multiplying the term frequency values with the logarithm of the inverse document frequency (IDF) [16].

The Word Vector Tool plugin[3] for YALE provides operators for pre-processing operations as stop word removal, word stemming, document vector generation, term weighting, etc. It allows to integrate sets of texts into YALE operator chains just like any other kind of example sets that can be described by attribute value pair vectors.

A particular problem for text mining are drifting concepts. Text data might be collected over an extended period of time and the target concept to be learned from the data stream may change. An important example is personalized information filtering, i.e. the adaptive classification of documents with respect to a dynamic user interest.

The concept drift plugin for YALE enables YALE to perform data stream mining and concept drift tracking. This plugin provides operators to simulate data streams and concept drift on arbitrary data sets as well as to handle real concept drift on real-world data. The methods range from totally ignoring the drift, learning with complete or no memory of old examples, and using time windows of fixed or adaptive size [9]. It also provides other example selection and weighting strategies [8] as well as variants of advanced boosting-like ensemble methods for handling concept drift [19] derived from those described in the previous section.

YALE allows the seamless integration of all the aforementioned steps from text data pre-processing via data stream simulation and handling, classifier learning, and concept drift tracking to operator chain and parameter variation and optimization as well as result visualization. The Word Vector Tool and the concept drift plugin are the basis for many other text mining applications, e.g. text clustering or keyword extraction.

[3] `http://wvtool.sf.net/`

3.3 Case study 3: Distributed data mining and feature construction

Distributed computing plays an important role in the data mining process for several reasons. First, Data Mining often requires huge amounts of resources in storage space and computation time. To make systems scalable, it is important to develop mechanisms that distribute the work load among several sites in a flexible way. Second, data is often inherently distributed into several databases, making a centralized processing of this data very inefficient and prone to security risks.

The YALE distributed data mining plugin allows to perform distributed data mining experiments in a simple and flexible way. The experiments are not actually executed on distributed network nodes. The plugin only simulates this. Simulation makes it easy to experiment with diverse network structures and communication patterns. Optimal methods and parameters can be identified efficiently before putting the system into use. The network structure can for example be optimized as part of the general parameter optimization. While this cannot replace testing the system in an actual network, it makes the development stage much more efficient. This follows the general philosophy of YALE as rapid prototyping tool for large scale data mining applications.

YALE was used for prototyping the distributed multimedia organization system NEMOZ[4]. Choosing the right representations of examples and hypotheses is essential to enable successful multimedia mining. Feature construction is an approach to find such a representation independently of the underlying learning algorithm. Unfortunately, the construction of features usually implies searching a very large space of possibilities and is often computationally demanding. Therefore, we proposed an approach to feature construction that is based on transferring information between different data mining tasks [11]. Tasks are stored together with a corresponding set of constructed features in a distributed case base. The case base is then used to constrain and guide the feature construction for new tasks. This is achieved by a new representation model for data mining tasks and a corresponding highly efficient distance measure. This approach is unique as it enables us to apply feature construction not only on a large scale, but also in scenarios in which communication cost plays an important role.

This kind of simulation is a good example for the power of the YALE data model. Different nodes share the same underlying database with different views. Hence, data and attribute information does not need to be replicated. Sharing examples and attributes among nodes is simply accomplished by "selecting" them. A freely definable cost matrix allows to account communication cost and time in a very flexible way. The same holds for the network structure.

3.4 Case study 4: Feature selection for unsupervised learning

The aim of cluster analysis is to group data points into natural clusters of similar data points. Such natural clusters describe the original data space in a structured way. They can be however covered by noisy features, sparsity or redundant features. Feature space transformation is therefore an important preprocessing step for many clustering tasks. In a current project, we explore possibilities to apply feature selection and transformation to unsupervised learning [12].

Feature space transformation for unsupervised learning is inherently a multi-objective optimization problem. On the one hand, the quality of the resulting clusters should be improved. On the other hand, we want to preserve as much of the original data space as possible, such that the discovered clusters are a valid representation of the original data. We use the capabilities of YALE to apply multi-objective feature space optimization as wrapper approach and combine it with several clustering algorithms as well as feature selection and construction schemes. All points of the resulting Pareto set describe optimal solutions concerning cluster quality *and* minimal transformation of the original data.

Multi-objective optimization requires usually a large number of individuals and generations. YALE enables such large scale experiments as the internal data management does not duplicate any data points. Again, the individual feature sets are merely different views on the same data.

3.5 Additional applications

Other applications of YALE include the prediction of parameters for chemical processes [5] and learning the preprocessing of time series data [10]. The natural language processing and text mining architecture GATE [4] has an internal Machine Learning API which will be based on YALE in a future version. MUSICMINER[5] is a music browser which extracts the necessary audio features by using the value series plugin of YALE.

4. RELATED WORK

A look at the KDnuggets software directory[6] reveals that a host of commercial data mining tools exists but only a few are freely available on an open source basis. Among the commercial tools, leading mining suites such as SPSS Clementine[7] and others usually offer a flat workflow-oriented view on the operations, using no nestable operators, and including a fairly limited range of standard learning algorithms. Tasks such as cross validation, parameter optimization or automatic feature selection are rarely directly supported. This situation is similar concerning the free software, but here at least developers may choose to extend the functionality since the source code is available.

Probably the most well-known open source data mining package is Weka [20], a collection of Java implementations of machine learning algorithms. All Weka algorithms are directly and seamlessly available in YALE. Other open source projects can be classified according to whether they provide a GUI with an integrated data flow model (such as Weka or Tanagra[8]), or consist of programming libraries like ADaM[9].

The preparation of data is supported in YALE by numerous feature selection and construction operators. However, YALE is applied to a single input data table. The MiningMart software [13, 6] can be used to combine data from several tables, or to prepare large data sets inside a relational database instead of main memory as in YALE. MiningMart provides operators that ease the integration with YALE.

YALE's XML-based experiment storage format can be seen

[4] http://nemoz.sf.net

[5] http://musicminer.sf.net

[6] http://www.kdnuggets.com/software/

[7] http://www.spss.com/clementine/

[8] http://chirouble.univ-lyon2.fr/~ricco/tanagra/

[9] http://datamining.itsc.uah.edu/adam/

as a language to model data mining processes which is interpreted by the YALE system. A number of other approaches use modeling languages for the mining process. MiningMart (see above) uses M4 which includes, besides modeling the mining process, a two-layered data meta model and can be stored both in a relational database or in XML files [7, 6]. KDDML [15] has been developed as a middleware language for the support of KDD applications. Elements in KDDML are operators with functional semantics; this allows to nest operators like in YALE, though YALE uses procedural semantics. Similarly, modeling languages for KDD processes have also been developed in research on KDD over grids; see [3] for an overview of this area. Two XML-based languages being developed in this context are DPML [1] and DSCL [2]. The XML-based syntax used by YALE is suitable for distributed processing as well; this is a direction for future work. Finally, the new PMML version 3.0[10] [14] includes facilities to model data transformations executed on a data set before mining. PMML is not process-oriented but provides a standard to describe machine-learned models in XML. Future work will integrate PMML import and export facilities into the YALE environment.

5. CONCLUSION

This paper presented YALE, a prototyping tool for complex KD tasks. YALE offers a wide variety of different algorithms and methods, which can be flexibly combined and arbitrarily nested. KD processes are represented as operator trees. This enables users to easily incorporate loop facilities into their KD experiments. Loops are essential for many tasks like parameter optimization, feature selection, or applying iterative learning schemes as boosting. YALE provides an internal data management system that allows arbitrary views on the data, without duplicating it. This is essential for large scale feature construction and the simulation of distributed data mining. This data management also keeps the data handling as transparent as possible to end users as well as to operator developers. Finally, YALE is easy to extend and many plugins already exist that enrich its base functionality. These plugins currently cover text, audio, time series, and multimedia processing, data stream simulation and concept drift handling, clustering, and distributed data mining.

YALE is currently used in more than 20 countries worldwide for research and development as well as real-world applications in companies as well as for research and teaching at universities. During the last twelve months, there were more than 40,000 downloads from the YALE web page[11] and more than 160,000 web page visits. Encouraged by this interest in our framework, we hope that in the future even more people will find YALE useful for their professional and academic work.

6. REFERENCES

[1] S. AlSairafi, F.-S. Emmanouil, M. Ghanem, N. Giannadakis, Y. Guo, D. Kalaitzopoulos, M. Osmond, A. Rowe, J. Syed, and P. Wendel. The Design of Discovery Net: Towards Open Grid Services for Knowledge Discovery. *High-Performance Computing Applications*, 17(3):297–315, 2003.

[2] P. Brezany, I. Janciak, A. Wöhrer, and A. M. Tjoa. GridMiner: A Framework for Knowledge Discovery on the Grid – from a Vision to Design and Implementation. In *Proceedings of the Cracow Grid Workshop*, Cracow, Poland, 2004.

[3] M. Cannataro, A. Congiusta, C. Mastroianni, A. Pugliese, D. Talia, and P. Trunfio. Grid-Based Data Mining and Knowledge Discovery. In N. Zhong and J. Liu, editors, *Intelligent Technologies for Information Analysis*. Springer, Berlin, Germany, 2004.

[4] H. Cunningham, K. Humphreys, R. Gaizauskas, and Y. Wilks. Software infrastructure for natural language processing. In *Proceedings of the Fifth Conference on Applied Natural Language Processing (ANLP-97)*, pages 237–244, San Francisco, CA, USA, 1997. Morgan Kaufmann Publishers Inc. http://gate.ac.uk/.

[5] G. Daniel, J. Dienstuhl, S. Engell, S. Felske, K. Goser, R. Klinkenberg, K. Morik, O. Ritthoff, and H. Schmidt-Traub. *Advances in Computational Intelligence – Theory and Practice*, chapter Novel Learning Tasks, Optimization, and Their Application, pages 245–318. Springer, 2002.

[6] T. Euler. Publishing Operational Models of Data Mining Case Studies. In *Proceedings of the Workshop on Data Mining Case Studies at the 5th IEEE International Conference on Data Mining (ICDM)*, pages 99–106, Houston, Texas, USA, 2005.

[7] J.-U. Kietz, A. Vaduva, and R. Zücker. MiningMart: Metadata-Driven Preprocessing. In *Proceedings of the ECML/PKDD Workshop on Database Support for KDD*, September 2001.

[8] R. Klinkenberg. Learning drifting concepts: Example selection vs. example weighting. *Intelligent Data Analysis (IDA), Special Issue on Incremental Learning Systems Capable of Dealing with Concept Drift*, 8(3):281–300, May 2004.

[9] R. Klinkenberg and T. Joachims. Detecting concept drift with support vector machines. In P. Langley, editor, *Proceedings of the Seventeenth International Conference on Machine Learning (ICML)*, pages 487–494, San Francisco, CA, USA, 2000. Morgan Kaufmann.

[10] I. Mierswa and K. Morik. Automatic feature extraction for classifying audio data. *Machine Learning Journal*, 58:127–149, 2005.

[11] I. Mierswa and M. Wurst. Efficient feature construction by meta learning – guiding the search in meta hypothesis space. In *Proc. of the Internation Conference on Machine Learning, Workshop on Meta Learning*, 2005.

[12] I. Mierswa and M. Wurst. Information preserving multi-objective feature selection for unsupervised learning. In *Proc. of the Genetic and Evolutionary Computation Conference (GECCO '06)*, 2006. submitted.

[13] K. Morik and M. Scholz. The MiningMart Approach to Knowledge Discovery in Databases. In N. Zhong and J. Liu, editors, *Intelligent Technologies for Information Analysis*, chapter 3, pages 47–65. Springer, Berlin, Germany, 2004.

[14] S. Raspl. PMML Version 3.0—Overview and Status. In R. Grossman, editor, *Proceedings of the Workshop on Data Mining Standards, Services and Platforms at the 10th ACM SIGKDD Int. Conf. on Knowledge Discovery and Data Mining (KDD)*, pages 18–22, 2004.

[15] A. Romei, S. Ruggieri, and F. Turini. KDDML: A Middleware Language and System for Knowledge Discovery in Databases. In *Proceedings of the 13th Italian Symposium on Advanced Database Systems (SEBD)*, June 2005.

[16] G. Salton and C. Buckley. Term weighting approaches in automated text retrieval. *Information Processing and Management*, 24(5):513–523, 1988.

[17] G. Salton, A. Wong, and C. S. Yang. A vector space model for automatic indexing. *Communications of the ACM (CACM)*, 18(11):613–620, November 1975.

[18] M. Scholz. Sampling-Based Sequential Subgroup Mining. In R. L. Grossman, R. Bayardo, K. Bennett, and J. Vaidya, editors, *Proceedings of the 11th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '05)*, pages 265–274, Chicago, Illinois, USA, August 2005. ACM Press.

[19] M. Scholz and R. Klinkenberg. Boosting Classifiers for Drifting Concepts. *Intelligent Data Analysis (IDA), Special Issue on Knowledge Discovery from Data Streams*, 2006. Accepted for publication.

[20] I. H. Witten and E. Frank. *Data mining: Practical machine learning tools and techniques with Java implementations*. Morgan Kaufmann, San Francisco, CA, USA, 2000. `http://www.cs.waikato.ac.nz/ml/weka/`.

[10] `http://www.dmg.org/pmml-v3-0.html`

[11] `http://yale.sf.net`