KNIME: The Konstanz Information Miner

Michael R. Berthold, Nicolas Cebron, Fabian Dill, Thomas R. Gabriel, Tobias Kötter, Thorsten Meinl, Peter Ohl, Christoph Sieb, Kilian Thiel and Bernd Wiswedel

ALTANA Chair for Bioinformatics and Information Mining,
Department of Computer and Information Science, University of Konstanz,
Box M712, 78457 Konstanz, Germany
contact@knime.org

Abstract. The Konstanz Information Miner is a modular environment, which enables easy visual assembly and interactive execution of a data pipeline. It is designed as a teaching, research and collaboration platform, which enables simple integration of new algorithms and tools as well as data manipulation or visualization methods in the form of new modules or nodes. In this paper we describe some of the design aspects of the underlying architecture and briefly sketch how new nodes can be incorporated.

1 Overview

The need for modular data analysis environments has increased dramatically over the past years. In order to make use of the vast variety of data analysis methods around, it is essential that such an environment is easy and intuitive to use, allows for quick and interactive changes to the analysis process and enables the user to visually explore the results. To meet these challenges data pipelining environments have gathered incredible momentum over the past years. Some of today's well-established (but unfortunately also commercial) data pipelining tools are InforSense KDE (InforSense Ltd.), Insightful Miner (Insightful Corporation), and Pipeline Pilot (SciTegic). These environments allow the user to visually assemble and adapt the analysis flow from standardized building blocks, which are then connected through pipes carrying data or models. An additional advantage of these systems is the intuitive, graphical way to document what has been done.

KNIME, the Konstanz Information Miner provides such a pipelining environment. Figure 1 shows a screenshot of an example analysis flow. In the center, a flow is reading in data from two sources and processes it in several, parallel analysis flows, consisting of preprocessing, modeling, and visualization nodes. On the left a repository of nodes is shown. From this large variety of nodes, one can select data sources, data preprocessing steps, model building algorithms, as well as visualization tools and drag them onto the workbench, where they can be connected to other nodes. The

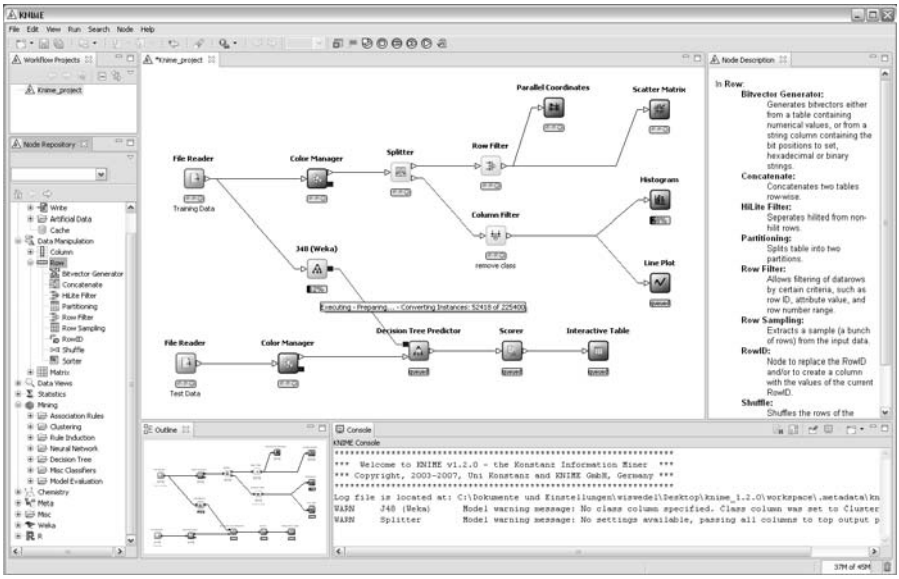

Fig. 1. An example analysis flow inside KNIME.

ability to have all views interact graphically (*visual brushing*) creates a powerful environment to visually explore the data sets at hand. KNIME is written in Java and its graphical workflow editor is implemented as an Eclipse (Eclipse Foundation (2007)) plug-in. It is easy to extend through an open API and a data abstraction framework, which allows for new nodes to be quickly added in a well-defined way.

In this paper we describe some of the internals of KNIME in more detail. More information as well as downloads can be found at `http://www.knime.org`.

2 Architecture

The architecture of KNIME was designed with three main principles in mind.

- Visual, interactive framework: Data flows should be combined by simple drag&drop from a variety of processing units. Customized applications can be modeled through individual data pipelines.
- Modularity: Processing units and data containers should not depend on each other in order to enable easy distribution of computation and allow for independent development of different algorithms. Data types are encapsulated, that is, no types are predefined, new types can easily be added bringing along type specific renderers and comparators. New types can be declared compatible to existing types.
- Easy expandability: It should be easy to add new processing nodes or views and distribute them through a simple plugin mechanism without the need for complicated install/deinstall procedures.

In order to achieve this, a data analysis process consists of a pipeline of nodes, connected by edges that transport either data or models. Each node processes the arriving data and/or model(s) and produces results on its outputs when requested. Figure 2 schematically illustrates this process. The type of processing ranges from basic data operations such as filtering or merging to simple statistical functions, such as computations of mean, standard deviation or linear regression coefficients to computation intensive data modeling operators (clustering, decision trees, neural networks, to name just a few). In addition, most of the modeling nodes allow for an interactive exploration of their results through accompanying views. In the following we will briefly describe the underlying schemata of data, node, workflow management and how the interactive views communicate.

2.1 Data structures

All data flowing between nodes is wrapped within a class called `DataTable`, which holds meta-information concerning the type of its columns in addition to the actual data. The data can be accessed by iterating over instances of `DataRow`. Each row contains a unique identifier (or primary key) and a specific number of `DataCell` objects, which hold the actual data. The reason to avoid access by Row ID or index is scalability, that is, the desire to be able to process large amounts of data and therefore not be forced to keep all of the rows in memory for fast random access. KNIME employs a powerful caching strategy which moves parts of a data table to the hard drive if it becomes too large. Figure 3 shows a UML diagram of the main underlying data structure.

2.2 Nodes

Nodes in KNIME are the most general processing units and usually resemble one node in the visual workflow representation. The class `Node` wraps all functionality and

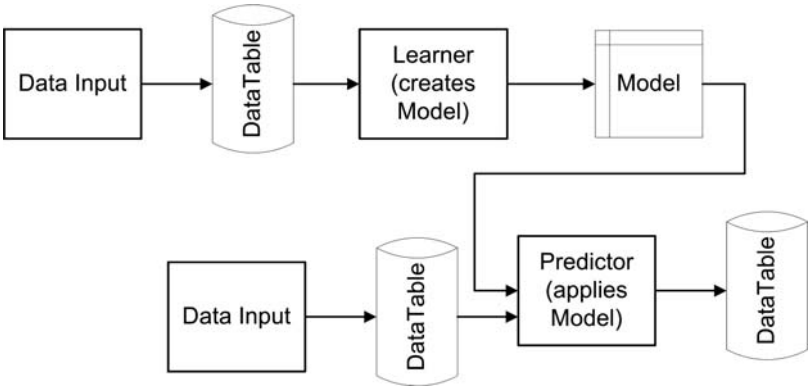

Fig. 2. A schematic for the flow of data and models in a KNIME workflow.

makes use of user defined implementations of a `NodeModel`, possibly a `NodeDialog`, and one or more `NodeView` instances if appropriate. Neither dialog nor view must be implemented if no user settings or views are needed. This schema follows the well-known Model-View-Controller design pattern. In addition, for the input and output connections, each node has a number of `Inport` and `Outport` instances, which can either transport data or models. Figure 4 shows a UML diagram of this structure.

2.3 Workflow management

Workflows in KNIME are essentially graphs connecting nodes, or more formally, a direct acyclic graph (DAG). The `WorkflowManager` allows to insert new nodes and to add directed edges (connections) between two nodes. It also keeps track of the status of nodes (configured, executed, ...) and returns, on demand, a pool of executable nodes. This way the surrounding framework can freely distribute the workload among a couple of parallel threads or – in the future – even a distributed cluster of servers. Thanks to the underlying graph structure, the workflow manager is able to determine all nodes required to be executed along the paths leading to the node the user actually wants to execute.

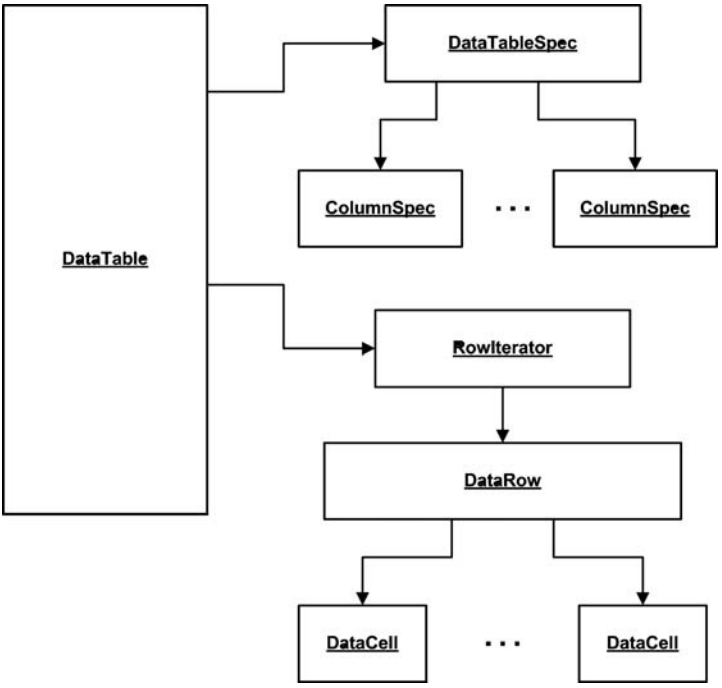

Fig. 3. A UML diagram of the data structure and the main classes it relies on.

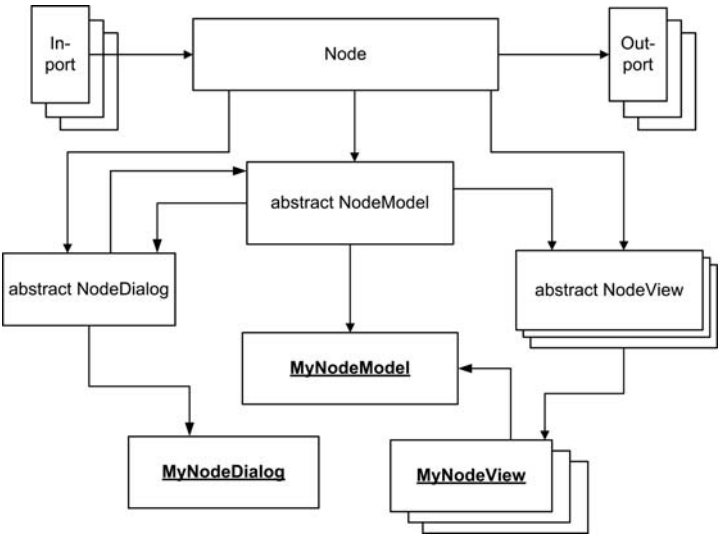

Fig. 4. A UML diagram of the Node and the main classes it relies on.

2.4 Views and interactive brushing

Each Node can have an arbitrary number of views associated with it. Through receiving events from a `HiLiteHandler` (and sending events to it) it is possible to mark selected points in such a view to enable visual brushing. Views can range from simple table views to more complex views on the underlying data (e. g. scatterplots, parallel coordinates) or the generated model (e. g. decision trees, rules).

2.5 Meta nodes

So-called *Meta Nodes* wrap a sub workflow into an encapsulating special node. This provides a series of advantages such as enabling the user to design much larger, more complex workflows and the encapsulation of specific actions. To this end some customized meta nodes are available, which allow for a repeated execution of the enclosed sub workflow, offering the ability to model more complex scenarios such as cross-validation, bagging and boosting, ensemble learning etc. Due to the modularity of KNIME, these techniques can then be applied virtually to any (learning) algorithm available in the repository.

Additionally, the concept of Meta Nodes helps to assign dedicated servers to this subflow or export the wrapped flow to other users as a predefined module.

2.6 Distributed processing

Due to the modular architecture it is easy to designate specific nodes to be run on separate machines. But to accommodate the increasing availability of multi-core ma-

chines, the support for shared memory parallelism also becomes increasingly important. KNIME offers a unified framework to parallelize data-parallel operations. Sieb et al. (2007) describe further extensions, which enable the distribution of complex tasks such as cross validation on a cluster or a GRID.

3 Repository

KNIME already offers a large variety of nodes, among them are nodes for various types of data I/O, manipulation, and transformation, as well as data mining and machine learning and a number of visualization components. Most of these nodes have been specifically developed for KNIME to enable tight integration with the framework; other nodes are wrappers, which integrate functionality from third party libraries. Some of these are summarized in the next section.

3.1 Standard nodes

- Data I/O: generic file reader, and reader for the attribute-relation file format (ARFF), database connector, CSV and ARFF writer, Excel spreadsheet writer
- Data manipulation: row and column filtering, data partitioning and sampling, sorting or random shuffling, data joiner and merger
- Data transformation: missing value replacer, matrix transposer, binners, nominal value generators
- Mining algorithms: clustering (k-means, sota, fuzzy c-means), decision tree, (fuzzy) rule induction, regression, subgroup and association rule mining, neural networks (probabilistic neural networks and multi-layer-perceptrons)
- Visualization: scatter plot, histogram, parallel coordinates, multidimensional scaling, rule plotters
- Misc: scripting nodes

3.2 External tools

KNIME integrates functionality of different open source projects that essentially cover all major areas of data analysis such as WEKA (Witten and Frank (2005)) for machine learning and data mining, the R environment (R Development core team (2007)) for statistical computations and graphics, and JFreeChart (Gilbert (2005)) for visualization.

- WEKA: essentially all algorithm implementations, for instance support vector machines, Bayes networks and Bayes classifier, decision tree learners
- R-project: console node to interactively execute R commands, basic R plotting node
- JFreeChart: various line, pie and histogram charts

The integration of these tools not only enriches the functionality available in KNIME but has also proven to be helpful to overcome compatibility limitations when the aim is on using these different libraries in a shared setup.

4 Extending KNIME

KNIME already includes plug-ins to incorporate existing data analysis tools. It is usually straightforward to create wrappers for external tools without having to modify these executables themselves. Adding new nodes to KNIME, also for native new operations, is easy. For this, one needs to extend three abstract classes:

- `NodeModel`: this class is responsible for the main computations. It requires to overwrite three main methods: `configure()`, `execute()`, and `reset()`. The first takes the meta information of the input tables and creates the definition of the output specification. The `execute` function performs the actual creation of the output data or models, and `reset` discards all intermediate results.
- `NodeDialog`: this class is used to specify the dialog that enables the user to adjust individual settings that affect the node's execution. A standardized set of `DefaultDialogComponent` objects allows the node developer to quickly create dialogs when only a few standard settings are needed.
- `NodeView`: this class can be extended multiple times to allow for different views onto the underlying model. Each view is automatically registered with a `HiLiteHandler` which sends events when other views have hilited points and allows to launch events in case points have been hilit inside this view.

In addition to the three model, dialog, and view classes the programmer also needs to provide a `NodeFactory`, which serves to create new instances of the above classes. The factory also provides names and other details such as the number of available views or a flag indicating absence or presence of a dialog.

A wizard integrated in the Eclipse-based development environment enables convenient generation of all required class bodies for a new node.

5 Conclusion

KNIME, the Konstanz Information Miner offers a modular framework, which provides a graphical workbench for visual assembly and interactive execution of data pipelines. It features a powerful and intuitive user interface, enables easy integration of new modules or nodes, and allows for interactive exploration of analysis results or trained models. In conjunction with the integration of powerful libraries such as the WEKA data mining toolkit and the R-statistics software, it constitutes a feature rich platform for various data analysis tasks.

KNIME is an open source project available at http://www.knime.org. The current release version 1.2.1 (as of 14 May 2007) has numerous improvements over the first public version released in July 2006. KNIME is actively maintained by a group of about 10 people and has more than 6 000 downloads so far. It is free for non-profit and academic use.

References

INFORSENSE LTD.: InforSense KDE. http://www.inforsense.com/kde.html.

INSIGHTFUL CORPORATION: Insightful Miner. http://www.insightful.com/products/iminer/default.asp.

SCITEGIC: Pipeline Pilot. http://www.scitegic.com/products/overview/.

ECLIPSE FOUNDATION (2007):*Eclipse 3.2 Documentation*. http://www.eclipse.org.

GILBERT, D. (2005): *JFreeChart Developer Guide*. Object Refinery Limited, Berkeley, California. http://www.jfree.org/jfreechart.

R DEVELOPMENT CORE TEAM (2007): *R: A language and environment for statistical computing*. R Foundation for Statistical Computing, Vienna, Austria. ISBN 3-900051-07-0. http://www.R-project.org.

SIEB C., MEINL T., and BERTHOLD, M. R. (2007): Parallel and distributed data pipelining with KNIME. *Mediterranean Journal of Computers and Networks, Special Issue on Data Mining Applications on Supercomputing and Grid Environments*. To appear.

WITTEN, I. H. and FRANK, E (2005): *Data Mining: Practical machine learning tools and techniques*. Morgan Kaufmann, San Francisco. http://www.cs.waikato.ac.nz/~ml/weka/index.html.