Open Source Data Pipelining für Interaktive Datenexploration

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

Universität Konstanz
ALTANA Lehrstuhl für Bioinformatik und Information Mining
Fachbereich Informatik und Informationswissenschaft
Fach M 712, 78457 Konstanz, contact@knime.org

Abstract

Der Konstanz Information Miner - KNIME - ist eine modulare Daten-Analyse Umgebung, die ein einfaches interaktives Erstellen und Ausführen datenfluss-orientierter Pipelines erlaubt. KNIME bietet als Lern-, Forschungs- und Kollaborations-Software eine ideale Plattform zur Anwendung von Daten-Transformations-, Visualisierungs- und Data-Mining Knoten. Durch seine erweiterbaren Schnittstellen ist es leicht möglich neue Algorithmen, aber auch bestehende Tools zu integrieren - u.a. sind Weka, das R-Project und CDK (Chemistry Development Kit) in KNIME verfügbar.

1 Überblick und Einleitung

Modulare Daten-Analyse-Plattformen kommen in den letzten Jahren immer mehr zur Anwendung. Um die große Auswahl von Analyse-Methoden nutzbar zu machen, ist es wichtig, dass solche Umgebungen einfach und intuitiv anzuwenden sind, schnelle und interaktive Änderungen im Analyse-Prozess erlauben und dem Benutzer ermöglichen, die Ergebnisse visuell zu explorieren. Diese Möglichkeiten führten dazu, dass Data Pipelines stark an Bedeutung gewonnen haben. Werkzeuge dieser Art ermöglichen es dem Benutzer Analyseabläufe aus standardisierten Verarbeitungseinheiten, die miteinander verbunden sind und durch die Daten oder Modelle fließen, visuell zusammenzusetzen und zu verändern. Ein weiterer Vorteil dieser Systeme ist die intuitive und graphische Art die einzelnen Analyseschritte nachzuvollziehen.

KNIME, der Konstanz Information Miner, bietet eine solche Pipeline Umgebung. Abbildung 1 zeigt den Screenshot eines Analyseablaufs. In der Mitte des Bildes ist ein Ablaufdiagramm zu erkennen: Es werden Daten von zwei Quellen eingelesen und in verschiedenen parallelen Zweigen, bestehend aus Daten-Vorverarbeitung, Modellbildung und Visualisierung verarbeitet. Eine Auswahl an Daten- und Modellverarbeitungsknoten sowie Visualisierungen ist auf der linken Seite zu sehen. Diese verschiedenen Module zum Einlesen von Daten-/Modellen, Vorverarbeitung, Modellbildung, Data-Mining-Algorithmen, sowie Visualisierung können leicht per Maus-Interaktion auf die Arbeitsfläche gezogen werden, wo sie mit anderen Knoten verbunden werden können.

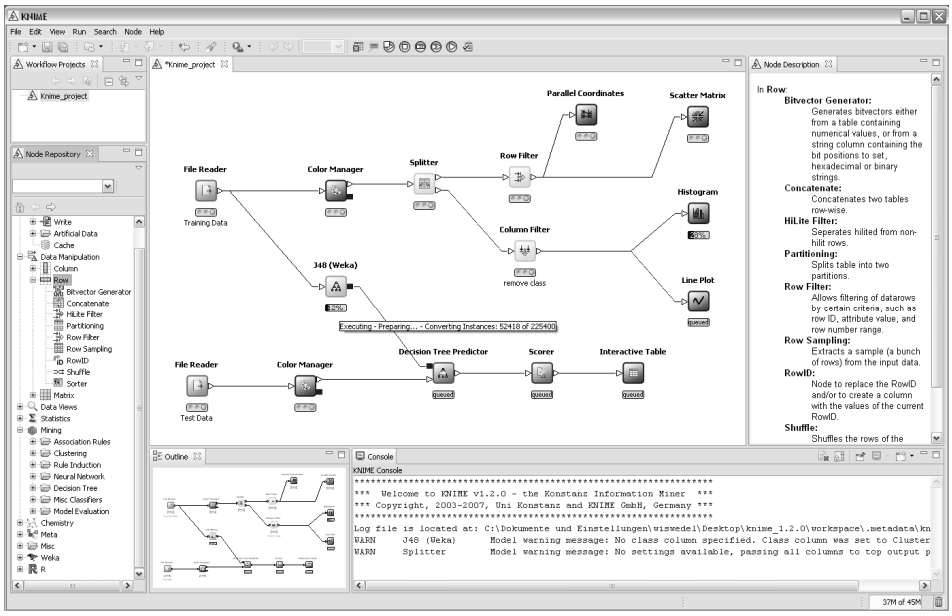

Abb. 1.: Ein Beispiel eines Analyseablaufs in KNIME.

Durch das visuelle Brushing (das graphische interagieren aller Visualisierungen) wird KNIME zu einer leistungsstarken und interaktiven Plattform um vorhandene Daten zu explorieren. KNIME ist in Java [Java07] geschrieben und seine graphische Arbeitsumgebung ist als Eclipse [EF07] Plug-in realisiert. Diese lässt sich durch offene Schnittstellen leicht mit neuen Knoten ergänzen.

Dieses Paper beschreibt die Architektur und die Besonderheiten von KNIME genauer. Für weitergehende Information sowie Downloads siehe http://www.knime.org.

2 Architektur

Die Architektur von KNIME basiert auf drei Prinzipien:

- Intuitives und interaktives Framework: Der Datenfluss lässt sich durch einfaches Drag&Drop aus verschiedenen Verarbeitungseinheiten (Knoten) erstellen. Benutzerspezifische Anwendungen können durch individuelle Data Pipelines modelliert werden.
- Modularität: Verarbeitungseinheiten und Daten-Container sind unabhängig und erlauben verteilte Berechnungen und dadurch die Entwicklung von neuen Algorithmen und Methoden in separaten Knoten. Datentypen sind abstrakt und ermöglichen leichtes Hinzufügen neuer Typen inklusive typen-spezifischer Renderer, Serialisierung und Komparatoren. Neue Typen lassen sich als kompatibel zu anderen, schon existierenden Typen deklarieren.

- Einfache Erweiterbarkeit: Das Hinzufügen neuer Verarbeitungsknoten oder Sichten ist durch die gegebene Programmstruktur extrem einfach. Durch den Eclipse Plug-in Mechanismus lassen sich diese dann leicht einbinden, ohne dass komplizierte Install-/Deinstallationsprozeduren notwendig sind.

Um das zu erreichen, besteht ein Daten-Analyseprozess aus einer Pipeline von Knoten, welche durch Kanten, die Daten oder Modelle transportieren, verbunden sind. Jeder Knoten verarbeitet die ankommenden Daten und/oder Modelle und gibt die Ergebnisse weiter an seine Ausgänge. Abbildung 2 zeigt eine schematische Darstellung dieses Ablaufs.

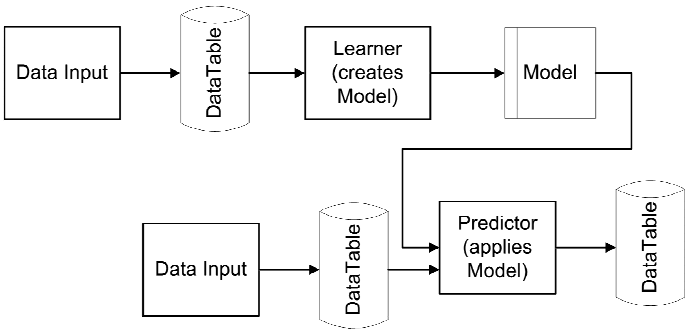

Abb. 2.: Skizze für einen Daten- und Modellfluss in KNIME.

Die Art der Verarbeitung erstreckt sich von grundlegenden Datenoperationen wie Filterung oder Zusammenführung, über einfache statistische Funktionen wie Berechnung des Mittelwertes, Standardabweichung oder lineare Regressionskoeffizienten, bis hin zu berechnungsintensiven Datenmodellierungsoperationen (Clustering, Entscheidungsbäume, Neuronale Netze, u.a.). Zusätzlich bieten die meisten Modellbildungsknoten die Möglichkeit, Ergebnisse in den zugehörigen Sichten interaktiv zu explorieren. Die folgenden Abschnitte beschreiben kurz die Datenstrukturen, die Knoten als Verarbeitungseinheiten und das Workflow-Management.

2.1 Datenstrukturen

Daten werden in KNIME in tabellarischer Form bzw. als Matrix mit einer vordefinierten Anzahl von Spalten und einer Menge von Zeilen gehalten. Jede Zeile und jede Spalte besitzt einen eindeutigen Bezeichner. Daten, die von einem zum nächsten Knoten fließen, werden in einem Objekt (`DataTable`) gespeichert, in welchem spezifische Metainformationen (`DataTableSpec` und `DataColumnSpec`) über die Spalten einer Tabelle, sowie die eigentlichen Daten gehalten werden. Der Zugriff auf die Daten geschieht per Iterator, der über alle Zeilen (`DataRow`) einer Tabelle läuft. Jede Zeile besteht aus einem eindeutigen Bezeichner (Primärer Key) und einer bestimmten Anzahl von `DataCell` Objekten, welche die eigentlichen Daten enthalten. Um Skalierbarkeit auf großen Datenmengen zu erreichen, ist der Zugriff auf Zeilen in

einer Tabelle nur per Iterator erlaubt, damit nicht alle Zeilen im Speicher gehalten werden müssen – direkter Datenzugriff per Index oder Zeilen ID ist somit nicht möglich. KNIME verwendet eine intelligente Caching-Strategie, d.h., Datenbestände, die zu groß werden, um sie im Hauptspeicher zu halten, werden auf die Festplatte ausgelagert und bei Bedarf wieder von dort gelesen. Abbildung 3 zeigt das Abhängigkeitsdiagramm der zugrunde liegenden Datenstruktur.

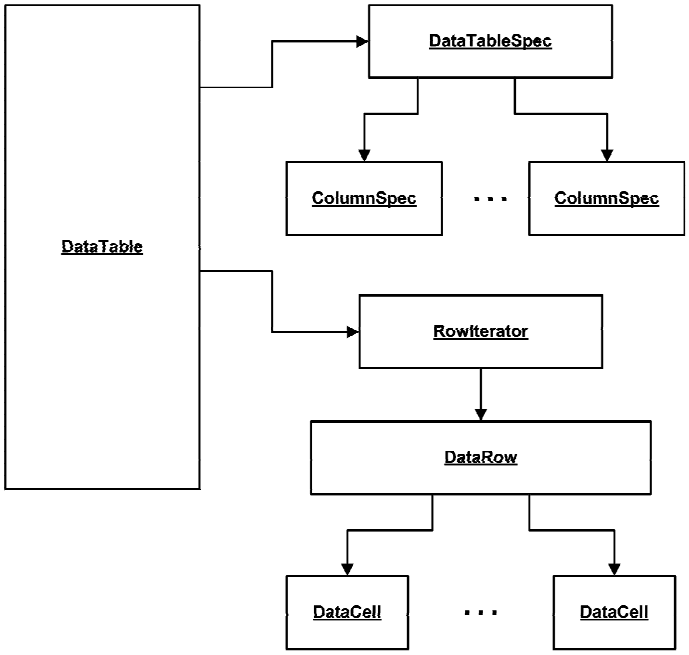

Abb. 3.: Das Diagramm der Tabelle (`DataTable`) und ihrer abhängigen Klassen.

2.2 Knoten als Verarbeitungseinheiten

Knoten in KNIME sind allgemeine Verarbeitungseinheiten und repräsentieren gewöhnlich ein bestimmtes Modul in einem Workflow. Die Klasse `Node` kapselt die komplette Funktionalität und bedient sich der benutzerspezifischen Implementierungen von `NodeModel` (dem Modell), `NodeDialogPane` (dem Dialog) und einer oder mehrerer `NodeView` Instanzen (den Sichten). Weder Dialog noch Sichten müssen implementiert werden, wenn keine benutzerdefinierten Einstellungen oder Darstellungen für einen bestimmten Knoten benötigt werden. Dieses Schema folgt dem bekannten Model-View-Controller Prinzip. Des Weiteren besitzt jeder Knoten Verbindungsstellen (Ports) für die Ein- und Ausgabe, über welche die Daten und/oder Modelle in den Knoten geleitet bzw. nach außen gegeben werden. Abbildung 4 stellt diese Struktur schematisch dar.

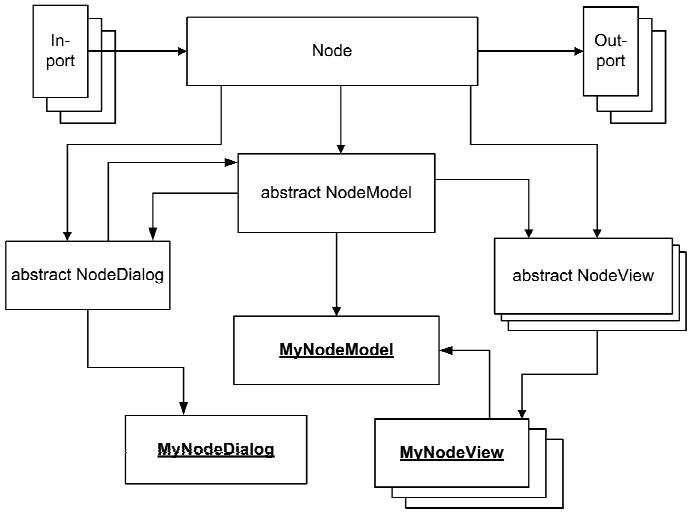

Abb. 4.: Diagramm eines Knotens (Node) und seiner abhängigen Klassen.

2.3 Workflow-Management

Workflows in KNIME sind azyklische Graphen, deren Knoten durch gerichtete Kanten verbundenen sind. Der `WorkflowManager`, als zentrale Klasse, erlaubt das Einfügen neuer Knoten und Verbindungen zwischen Knoten, die zum Workflow hinzugefügt werden sollen. Diese Klasse verwaltet den Status (konfiguriert, ausgeführt etc.) aller Knoten und liefert auf Anfrage einen Pool von ausführbaren Knoten. Auf diese Art kann das umgebende System die Ausführung auf parallel arbeitende Threads oder sogar auf Cluster verteilen. Durch die zugrunde liegende Graphstruktur des Workflows ist es für den `WorkflowManager` einfach, alle ausführbaren Knoten, abhängig von der aktuellen Knotenselektion, zu ermitteln.

2.4 Sichten und Interaktives Brushing

Jeder Knoten kann eine beliebige Anzahl von zugehörigen Sichten (Views) auf Daten und/oder Modelle haben. Durch die Möglichkeit Events über den `HiLiteHandler` zwischen Knoten und damit registrierten Sichten zu senden, können Views miteinander kommunizieren. D.h., Datenpunkte, die in einer Sicht markiert wurden, werden in einer anderen Datendarstellung ebenfalls hervorgehoben und erlauben somit interaktives Brushing. Sichten reichen von einfachen tabellarischen Darstellungen bis hin zu komplexen Ansichten auf die zugrunde liegenden Daten (z.B. 2D Scatterplot, Parallele Koordinaten, usw.) oder Modelle (z.B. Entscheidungsbäume, Regeln, usw.). Eine weitere Stärke von KNIME ist die Möglichkeit Hervorhebungen zwischen Daten und Modellen aufzulösen, d.h., durch Hervorheben bestimmter Modellteile werden die damit erklärten Datenpunkte ebenfalls markiert.

2.5 Meta-Knoten

Die so genannten Meta-Knoten erlauben es Teile eines Workflows zu einem Knoten zusammenzufassen, der wiederum die gleichen Eigenschaften wie ein Standardknoten besitzt. Zusätzlich bietet dieser Knoten eine Reihe von Vorteilen: Der Benutzer ist in der Lage größere und komplexere Workflows übersichtlich zu gestalten und bestimmte Aktionen in einem Knoten zusammenzufassen.

In KNIME existieren vordefinierte Meta-Knoten, die das mehrmalige Ausführen von Teilworkflows erlauben, zum Beispiel komplexe Szenarien wie Cross-Validierung, Bagging, Boosting, Ensemble Lernen etc. Durch die Modularität von KNIME ist es möglich diese Techniken auf quasi beliebige (Lern-) Algorithmen anzuwenden.

Zusätzlich hilft das Meta-Knoten-Konzept Teilworkflows auf bestimmte Server auszulagern oder sie anderen Benutzern als vordefinierte Module zur Verfügung zu stellen.

2.6 Verteilte Verarbeitung

Die modulare Architektur ermöglicht es spezifische Knoten auf anderen Maschinen auszuführen. Um auf der anderen Seite mit der zunehmenden Entwicklung von Multi-Prozessor-Computern mitzuhalten, ist die Unterstützung von Parallelität auf gemeinsam genutztem Speicher ebenso wichtig. KNIME bietet ein einheitliches Framework um Datenoperationen parallel auszuführen. Sieb et al. (2007) beschreiben eine Erweiterung basierend auf KNIME, um komplexe Aufgaben wie Cross-Validierung auf einem Cluster oder GRID parallel auszuführen.

3 Knoten-Palette

In KNIME existieren in der Standard-Version verschiedene Knoten für Daten I/O, Manipulationen und Transformationen, sowie Data-Mining, Methoden des Machine-Learning und Visualisierungskomponenten. Viele dieser Knoten wurden speziell für KNIME entwickelt und sind bestens auf das Framework abgestimmt; andere kapseln Funktionalität von Drittanbietern. Der nächste Abschnitt listet einige dieser Knoten auf.

3.1 Standard Knoten

- Daten I/O: Knoten für: ASCII-Dateien, CSV, ARFF Format, Datenbanken
- Daten-Manipulation: Filtern von Zeilen- und Spalten, Partitionieren und Auswählen, zufälliges Umsortieren, Zusammenführen, Vermischen
- Daten-Transformation: Ersetzen fehlender Werte, Matrix-Transponieren, Klasseneinteilen (Binning), Wertebereichsfinder
- Mining-Algorithmen: Clustering (k-Means, SOTA, fuzzy c-Means), Entscheidungsbäume, (Fuzzy) Regeln, Regression, Subgroup- und Assoziationsregel-Mining, Neuronale Netze (PNN und MLP)

- Visualisierungen: 2D Scatterplot, Histogramm, Parallele Koordinaten, multi-dimensionale Skalierung, Regel- und Entscheidungsbaum-Visualisierung
- Sonstige: Java-Code (Skripting) Knoten

3.2 Externe Tools

In KNIME sind verschiedene Open-Source-Projekte integriert, die große Teile der Daten-Analyse abdecken wie WEKA [WiFr05] für Machine-Learning und Data-Mining, die R-Library [R05] für statistische Berechnungen und Visualisierungen, JFreeChart [Gilb05] für Visualisierungen und CDK [StHK+03] für chemo- und bioinformatische Berechnungen und Visualisierungen.

- WEKA: alle Algorithmen sind in KNIME verfügbar, darunter Support Vektor Maschine, Bayes Netzwerke, Bayes-Klassifizierer, Entscheidungsbaumlerner
- R-Project: Konsole und Skripting-Knoten um Befehle direkt an R zu senden
- JFreeChart: Histogramm, Linien- und Tortendiagramm
- CDK: Java Open-Source -Software für Chemo- und Bioinformatik

Die Integration dieser Werkzeuge bereichert nicht nur die Funktionalität von KNIME, sondern zeigt auch wie nützlich es sein kann Kompatibilitätsprobleme zu lösen, um unterschiedliche Bibliotheken in einer gemeinsamen Umgebung zu verschmelzen.

4 KNIME Erweiterungen

Neue Knoten können leicht für KNIME entwickelt und über den Eclipse Plug-in Mechanismus integriert werden. Hierfür stehen offene Java-Schnittstellen zur Verfügung, um zum Beispiel eigene Anwendungen (auch anderer Programmier-sprachen) zu kapseln, ohne den bestehenden Code anpassen zu müssen. Die drei Basisklassen `NodeModel`, `NodeDialogPane` und `NodeView` liefern das Grundgerüst für einen neuen Knoten:

- `NodeModel`: Diese Klasse ist für die Verarbeitung der Eingabe zuständig, berechnet die Ausgabe und interagiert mit dem Dialog und der View. Die drei wichtigsten Methoden, die implementiert werden müssen, sind: `configure()`, `execute()` und `reset()`: Die erste nimmt die Metainformationen der Eingabetabellen, validiert gegen diese die Einstellungen aus dem Dialog und liefert, wenn möglich, die Spezifikation der Ausgabetabellen. Die `execute()`-Methode berechnet die Ausgabetabelle oder das Datenmodell anhand der Eingabedaten, während im `reset()` alle lokalen Modellvariablen zurückgesetzt werden müssen.
- `NodeDialogPane`: Innerhalb des Dialoges können GUI-Komponenten angezeigt werden, die es erlauben Einstellungen vorzunehmen, die dann im Knoten während der Ausführung ausgewertet werden. Vordefinierte `DefaultDialogComponent` Objekte ermöglichen es, schnell eigene Dialoge für Parameter mit Standardtypen zu erstellen.

- `NodeView`: Zu jedem Knoten kann eine beliebige Anzahl von Views registriert werden, die unterschiedliche Sichten auf das Modell bieten. Jede dieser Sichten ist automatisch mit einem `HiLiteHandler` verbunden, um Hervorhebungen anderer verbundener Views darzustellen und Markierungen in anderen Views entsprechend zu veranlassen.

Zusätzlich zu diesen drei Klassen ist es notwendig, eine Klasse `NodeFactory` anzulegen, die als Erzeuger obiger Basisklassen agiert. Die Factory-Klasse liefert weitere Information bezüglich des Knotens, wie Namen, Anzahl der Views, Dialog, sowie eine Beschreibung. Ein in die Eclipse Entwicklungsumgebung integrierter Wizard steht zur Verfügung, um alle notwendigen Klassen zu erzeugen und den Knoten im Framework anzumelden.

5 Zusammenfassung

KNIME, der Konstanz Information Miner, ist eine modulare, in eine graphischen Umgebung eingebettete Plattform, die visuelles und interaktives Ausführen von datenflussorientierten Pipelines erlaubt. KNIME bietet eine leistungsfähige und intuitive Benutzerschnittstelle, erlaubt einfache Integration neuer Module oder Knoten und ermöglicht die interaktive Exploration der Analyseergebnisse oder trainierten Modelle. Zusammen mit der Integration umfangreicher Bibliotheken wie dem WEKA Data-Mining Toolkit und der R-Statistik Software, bildet es eine Plattform mit vielen Funktionen für viele Daten-Analyse-Anwendungen. KNIME ist ein Open-Source-Projekt und verfügbar unter http://www.knime.org. Die aktuelle Version 1.2.1 (vom 14. Mai 2007) beinhaltet eine Reihe von Verbesserungen gegenüber der ersten öffentlichen Release im Juli 2006. KNIME wird von mehr als 10 Leuten entwickelt und gepflegt und hat bis heute über 5500 Downloads. Die Software ist kostenlos für nicht-kommerzielle und akademische Zwecke.

Referenzen

[EcFo07] Eclipse Foundation: Eclipse 3.2 Documentation, 2007. http://www.eclipse.org

[Gilb05] JFreeChart Developer Guide: D. Gilbert, Object Refinery Limited, Berkeley, California. 2005. http://www.jfree.org/jfreechart

[Java07] Sun Developer Network (SDN), Java 5 (JDK), 2007. http://java.sun.com

[R05] R Development Core Team: R – A language and environment for statistical computation. R Foundation for Statistical Computing, Vienna, Austria, 2005. http://www.R-project.org

[SiMB07] C. Sieb, T. Meinl, and M. R. Berthold, Parallel and distributed data pipelining with KNIME. Mediterranean Journal of Computers and Networks, Special Issue on Data Mining Applications in Supercomputing and Grid Environments, 2007.

[StHK+03] C. Steinbeck, Y. Han, S. Kuhn, O. Horlacher, E. Luttmann, and E.L. Willighagen. The Chemistry Development Kit (CDK): an open-source Java library for Chemo- and Bioinformatics. J Chem Inf Comput Sci., 2003, http://cdk.sourceforge.net

[WiFr05] I. H. Witten, and E. Frank, Data Mining: Practical machine learning tools and techniques. Morgan Kaufmann, San Francisco, 2005. http://www.cs.waikato.ac.nz/~ml/weka/index.html