Knowing: A Generic Data Analysis Application

T. Bernecker, F. Graf, H.-P. Kriegel, N. Seiler
Institute for Informatics,
Ludwig-Maximilians-Universität München
Oettingenstraße 67, 80538 München
{bernecker,graf,kriegel}@dbs.ifi.lmu.de
nepomuk.seiler@campus.lmu.de

C. Türmer, D. Dill
Heinz-Nixdorf-Lehrstuhl für Medizinische Elektronik, TU München
Theresienstraße 90 / N3, 80333 München
{tuermer,dill}@tum.de

ABSTRACT

Extracting knowledge from data is, in most cases, not restricted to the analysis itself but accompanied by preparation and post-processing steps. Handling data coming directly from the source, e.g. a sensor, often requires preconditioning like parsing and removing irrelevant information before data mining algorithms can be applied to analyze the data. Stand-alone data mining frameworks in general do not provide such components since they require a specified input data format. Furthermore, they are often restricted to the available algorithms or a rapid integration of new algorithms for the purpose of quick testing is not possible. To address this shortcoming, we present the data analysis framework *Knowing*, which is easily extendible with additional algorithms by using an OSGi compliant architecture. In this demonstration, we apply the *Knowing* framework to a medical monitoring system recording physical activity. We use the data of 3D accelerometers to detect activities and perform data mining techniques and motion detection to classify and evaluate the quality and amount of physical activities. In the presented use case, patients and physicians can analyze the daily activity processes and perform long term data analysis by using an aggregated view of the results of the data mining process. Developers can integrate and evaluate newly developed algorithms and methods for data mining on the recorded database.

1. INTRODUCTION

Supporting the data mining process by tools was and still is a very important step in the history of data mining. By the support of several tools like ELKI [1], MOA [2], Weka [3] or RapidMiner [4], scientists are nowadays able to apply a diversity of well-known and established algorithms on their data for quick comparison and evaluation. In cases where the requirements enforce a rapid development from data mining to a representative prototype, these unstandardized plug-in systems can cause a significant delay which is caused by the time needed to incorporate the algorithms.

With the use of a standardized plug-in system like OSGi[1], Java Plugin Framework (JPF) or Java Simple Plugin Framework (JSPF), each implementation of an algorithm does not have to be specifically adapted to the according framework. With *Knowing* (*Know*ledge Engineer*ing*) we provide a framework that addresses this shortcoming by bridging the gap between the data mining process and rapid prototype development. We achieve this by using a standardized plug-in system based on OSGi, so that algorithms can be packed in OSGi resource bundles. This offers the possibility to either create new algorithms as well as to integrate and exchange existing algorithms from common data mining frameworks. The advantage of these OSGi compliant bundles is that they are not restricted for use in *Knowing* but can be used in any OSGi compliant architecture.

This demonstration includes the following contributions:

- a simple, yet powerful graphical user interface (GUI),
- a bundled embedded database as data storage,
- an extensible data mining functionality,
- extension support for algorithms addressing different use cases and
- a generic visualization of the results of the data mining process.

Section 2 provides a short overview of related work. Details of the architecture of *Knowing* will be given in Section 3. In our scenario, described in Section 4, we will present the *MedMon* system which itself extends *Knowing*. In the developer stage, we can easily switch between the scientific data mining view and the views which will be presented to the end users later on. As *MedMon* is intended to be used by different target groups (physicians and patients), it is desired to use a single base system for all views and only deploy different user interface bundles for each target group. This way, the data mining process can seamlessly be integrated into the development process by reducing long term maintenance to a minimum, as only a single system with different interface bundles has to be kept up to date and synchronized instead of a special data mining tool, a physician tool and a patient tool. Section 5 describes the planned demo tour in more detail.

[1]OSGi: http://en.wikipedia.org/wiki/OSGI

2. RELATED WORK

In the past years, several data mining frameworks like ELKI [1], MOA [2], WEKA [3], RapidMiner [4] or R [5] have been presented and established (among many others). Although all frameworks perform data mining in their core, they all have different target groups:

WEKA and MOA provide both algorithms and GUIs. By using these GUIs, the user can analyze data sets, configure and test algorithms and visualize the outcome of the according algorithm for evaluation purposes without needing to do some programming. As the GUI cannot satisfy all complex scenarios, the user still has the possibility to use the according APIs to build more complex scenarios in his own code. RapidMiner integrates WEKA and provides powerful analysis functionalities for analysis and reporting which are not covered by the WEKA GUI itself. RapidMiner also provides an improved GUI and also defines an API for user extensions. Both RapidMiner and WEKA provide some support to external databases. The aim of ELKI is to provide an extensible framework for different algorithms in the fields of clustering, outlier detection and indexing with the main focus on the comparability of algorithm performance. Therefore, single algorithms are not extensively tuned to performance but tuning is done on the application level for all algorithms and index structures. Like the other frameworks, ELKI also provides a GUI, so that programming is not needed for the most basic tasks. ELKI also provides an API that supports the integration of user-specified algorithms and index structures.

All the above frameworks provide support for the process of quick testing, evaluating and reporting and define APIs in different depths. Thus, scientists can incorporate new algorithms into the systems. However, none of them makes use of a standardized plug-in system, so that each implementation of an algorithm is specifically adapted to the according framework without being interchangeable.

R provides a rich toolbox for data analysis. Also there are a lot of plugins which extend the functionality of R. Nevertheless, all advantages and disadvantages like for the above frameworks also hold for R, especially the lack of a standardized plug-in system.

3. ARCHITECTURE

Applying a standardized plug-in system like OSGi, the bundles can be used in any OSGi compliant architecture like the Eclipse Rich Client Platform (RCP)[2] or the NetBeans RCP[3]. Then, the integration of existing algorithms can simply be done by wrapping and packing them into a separate bundle. Such bundles are then registered as independent service providers to the framework. In either case, algorithms are wrapped into Data Processing Units (DPU) which can be integrated and configured via pluggable RCP-based GUI controls. Thus, the user is able to perform an arbitrary amount of steps to pre- and post-process the data. Furthermore, we provide the possibility to use the DPUs contained in the system in any other OSGi compliant architecture. As dependencies between resource bundles have to be modeled explicitly, it is much easier to extract certain bundles from the system. This loose coupling is not only an advantage in case where algorithms should be ported between completely different systems, but also if the GUI should be changed from a data mining view to a prototype view for the productive system. This can be done by either using the resource bundles containing the DPUs, or by directly extending *Knowing* itself.

In the current implementation, the *Knowing* framework is based on the established and well-known Eclipse RCP system and uses the standardized OSGi architecture[4] which allows the composition of different bundles. This brings the great advantage that data miners and developers can take two different ways towards their individual goal: If they start a brand new RCP-based application, they can use *Knowing* out of the box and create the application directly on top of *Knowing*. The more common case might be that an RCP- or OSGi-based application already exists and should only be extended with data mining functionality. In this case, only the appropriate bundles are taken from *Knowing* and integrated into the application.

In the following, we describe the architecture of the *Knowing* framework which consists of a classical three-tier architecture comprising data storage tier, data mining tier and GUI tier, where each tier can be integrated or exchanged using a modular concept.

3.1 Data Storage

The data storage tier of *Knowing* provides the functionality and abstraction layers to access, import, convert and persist the source data. The data import is accomplished by an import wizard using service providers, so that importing data is not restricted to a certain format.

Applying the example of the *MedMon* application, a service provider is registered that reads binary data from a 3D accelerometer [7] which is connected via USB. The data storage currently defaults to an embedded Apache Derby database[5] which is accessed by the standardized Java Persistence API (JPA & EclipseLink). This has the advantage that the amount of data being read is not limited by the main memory of the used workstation and that the user does not have to set up a separate database server on his own. However, by using the JPA, there is the possibility to use more than 20 elaborated and well-known database systems which are supported by this API[6]. An important feature in the data storage tier arises from the possibility to use existing data to support the evaluation of newly recorded data, e.g. to apply certain parts of the data as training sets or reference results.

3.2 Data Mining

This tier includes all components needed for data mining and data analysis. OSGi bundles containing implemented algorithms are available fully transparently to the system after the bundle is registered as a service provider.

Algorithms are either implemented directly or wrapped in DPUs. Following the design of WEKA, DPUs represent filters, algorithms or classifiers. One or more DPUs can be bundled into an OSGi resource bundle which is registered into the program and thus made available in the framework. Bundling algorithms enforces a pluggable and modular architecture so that new algorithms can be integrated and re-

[2]Eclipse RCP: http://www.eclipse.org/platform/
[3]NetBeans RCP: http://netbeans.org/features/platform/
[4]Eclipse Equinox: http://www.eclipse.org/equinox/
[5]Apache Derby: http://db.apache.org/derby/
[6]List of supported databases:
http://wiki.eclipse.org/EclipseLink/FAQ/JPA

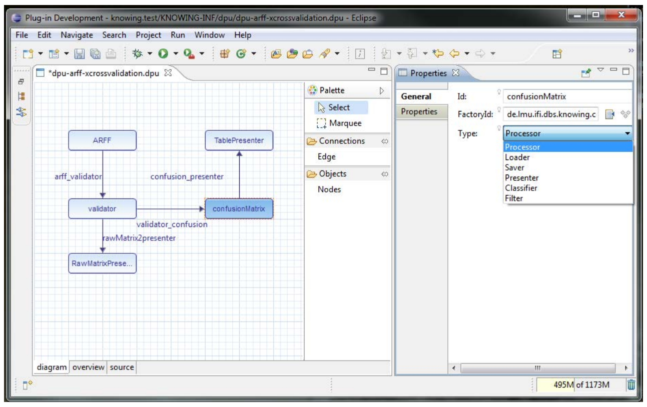


Figure 1: The *Knowing* user interface.

moved quickly without the need for extensive dependency checks. The separation into bundles also provides the possibility of visibility borders between bundles so that separate bundles remain independent and thus the system remains maintainable. The modularity also provides the possibility to concatenate different algorithms into processing chains so that algorithms can act both as sources and targets of processed entities (cf. Figure 1). Raw data for example first could pass one or more filtering components before being processed by a clustering component.

Creating a processing chain (a.k.a. model) of different, concatenated algorithms and data-conditioning filters is supported by GUI controls, so that different parameters or concatenations can be tested easily. After a model has proved to fit the needs of a use case, the model can be bundled and later be bound to other views of the GUI, so that the costs for porting, adapting and integration are minimized to binding components and models together. Hence, porting and adapting algorithms and other components from different APIs is not needed.

This architecture provides the possibility to integrate algorithms from other sources like [1, 2, 3, 5], so that existing knowledge can be reused without having to re-implement algorithms from scratch. This also provides the possibility to replace components by different implementations quickly if performance or licensing issues require to do so.

In the data mining part of the application, *Knowing* not only supports plain Java but also relies on the use of the Scala programming language. Scala is a functional and object-oriented programming language which is based on the Java Virtual Machine, so that it seamlessly integrates into *Knowing*. The advantage of Scala in this part of the application lies in the easy possibility of writing functional code shorter than in regular Java code. By using the Akka actor-model[7], it is easy to create processing chains which are executed in a parallel way so that *Knowing* can make use of multi-core systems.

3.3 User Interface

Using the well-established Eclipse RCP and its powerful concept of views enables developers to easily replace the view of the data mining scientists with different views for end users or prototypes. Thus, the task of porting data mining algorithms and the data model to the final application is replaced by just switching the view component and binding model and GUI components together. As Eclipse itself is designed as an RCP using OSGi, it is comparatively easy to unregister the original *Knowing* GUI and replace it with an interface representing the final application.

[7]Project Akka: http://akka.io/

4. DEMO PROGRAM

We motivate our demonstration by following a real-world use case where the convalescence of patients should be monitored by analyzing their daily physical activity as presented in the works of [6] and [7]. Among others, features like quality, intensity and amount of physical activity are diagnostically strongly conclusive as they have major influence on medical prevention, convalescence and therapy.

Physical activity in this case includes various types of motion like walking, running and cycling. The task is to perform data mining on long-term temporal sensor data provided by people wearing a little 3D sensor which is recording and storing acceleration data in all three axes with a frequency of 25 Hz. When the sensor is connected to a computer, the data is parsed and transferred to the *Knowing* framework, where it is stored in the underlying database. *Knowing* is able to deal with different types of time series which are not limited to the medical field but can be applied to different types of scenarios where time series data is being produced and needs to be analyzed. Analyzing the data in this use case means the application of clustering and classification techniques in order to detect motion patterns of activities and thus to separate the different types of motions. Available algorithms as well as additionally implemented techniques for data mining and the pre-conditioning of the temporal data (e.g. filtering of specific information, dimensionality reduction or removing noise) can efficiently be tested and evaluated on the data and can furthermore be applied to the data by taking advantage of the OSGi compliant architecture (cf. Section 3). By using the standardized OSGi plug-in system, we are integrating and embedding well-known data mining tools and, thus, avoid the re-implementation of already tested algorithms. The requirement of a quick migration of the final data mining process chain to a prototype system is accomplished by using different graphical views on a common platform. Thus, neither the process model nor the algorithms needs to be ported. Instead, only a different view of the same base model needs to be activated to enable the prototype. Finally, the demo provides a generic visualization model to present the results of the data mining process to the user.

5. DEMO TOUR

In this demo, we present an early stage of the application prototype *MedMon* (*Med*ical *Mon*itoring), which is based on *Knowing*. *MedMon* is a prototype of a use case for monitoring a patient's activity to support his/her convalescence [6, 7]. An exemplary GUI frame is depicted in Figure 2. By using the *MedMon* application, the users can import 3D acceleration data from the hardware sensor into a database. This is the first step of the demonstration: the sensor is worn as an electronic tag. So the user can go around and record some raw data that represent patients' activities as time series. The raw data is transmitted to the *MedMon* application by connecting the sensor to the computer. The import of the raw data is simplified by a wizard, which includes a preview of the time series. Working with *MedMon*, the user is enabled to switch between different roles. The prototype allows several views on the recorded data and the results of the data mining process:

- the *data mining view*, where DPUs can be combined to processing chains and which allows to employ newly developed algorithms;
- the *physician view*, which provides a more detailed view on the data for multiple users' activities and the possibility to add and modify electronic health records;
- and the *patient view*, which displays only a very brief summarization of the patient's daily activity in order to give feedback to the user about his achieved activity pensum each day.

In the presented use case, we can analyze the daily activity processes and perform long-term data analysis by using an aggregated view of the results of the data mining process from the physician view and the patient view. Presenting the data mining view in detail, we show the integration of newly developed algorithms and methods for data mining on the recorded database. Furthermore, we outline how to adjust the data mining process chain and to set the algorithm-specific parameters.

More precisely, the current process comprises the import of sensor data from binary files, followed by the segmentation of the data, the extraction of features including a linear discriminant analysis (LDA), the building of AR-models on the extracted segments and the classification of the results. Here, the demo user is able to decide whether to add noise filters for the raw data, select appropriate features to represent the segments or to choose from different classification or clustering methods.

In an evaluation example, we apply k-means clustering on the data with different parameters combined with a k-nearest neighbor classification to detect and classify different motion patterns. Here, recently recorded acceleration data created by the same patient could, for example, serve as training sets for the current evaluation. Finally, we show the possibilities to present the results in the most intuitive way in order to simulate an application scenario from a physician's perspective.

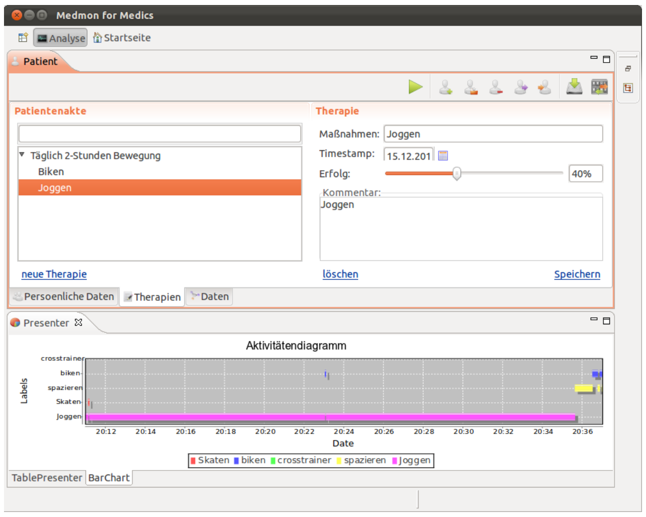
The *MedMon* prototype system is not limited to medical applications but provides a valuable tool for scientists having to deal with large amounts of time series data.

The source code of the *Knowing* framework, the *MedMon* prototype in its current state and the project wiki are available via GitHub[8].

6. CONCLUSION

In this demo, we present the open *Knowing* framework that allows faster integration of data mining techniques into the development process so that information and data can be managed more effectively. We show the integration of *Knowing* in the application of medical monitoring and outline the bridge between data mining and development. In future work, we will integrate more well-known data mining frameworks and extend the data mining GUI for faster testing of machine learning techniques.

[8] *Knowing* at GitHub: https://github.com/knowing



Maßnahmen: Joggen
Timestamp: 15.12.201
Erfolg: 40%
Kommentar: Joggen

Figure 2: The *MedMon* prototype GUI.

Acknowledgements

This research has been supported in part by the THESEUS program in the MEDICO and CTC projects. They are funded by the German Federal Ministry of Economics and Technology under the grant number 01MQ07020. The responsibility for this publication lies with the authors.

7. REFERENCES

[1] E. Achtert, T. Bernecker, H.-P. Kriegel, E. Schubert, and A. Zimek. ELKI in time: ELKI 0.2 for the performance evaluation of distance measures for time series. In *Proceedings of the 11th International Symposium on Spatial and Temporal Databases (SSTD), Aalborg, Denmark*, pages 436–440, 2009.

[2] A. Bifet, G. Holmes, R. Kirkby, and B. Pfahringer. MOA: Massive online analysis. 11:1601–1604, 2010.

[3] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten. The WEKA data mining software: an update. *ACM SIGKDD Explorations*, 11(1):10–18, 2009.

[4] I. Mierswa, M. Wurst, R. Klinkenberg, M. Scholz, and T. Euler. YALE: Rapid prototyping for complex data mining tasks. In *Proceedings of the 12th ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD), Philadelphia, PA*, pages 935–940, 2006.

[5] R project. `http://www.r-project.org/`.

[6] C. Türmer, D. Dill, A. Scholz, M. Gül, T. Bernecker, F. Graf, H.-P. Kriegel, and B. Wolf. Concept of a medical activity monitoring system improving the dialog between doctors and patients concerning preventions, diagnostics and therapies. In *Forum Medizin 21, Evidenzbasierte Medizin (EbM), Salzburg, Austria*, 2010.

[7] C. Türmer, D. Dill, A. Scholz, M. Gül, A. Stautner, T. Bernecker, F. Graf, and B. Wolf. Conceptual design for an activity monitoring system concerning medical applications using triaxial accelerometry. In *Austrian Society for Biomedical Engineering (BMT), Rostock, Germany*, 2010.