Using ADMIRE data mining and integration tools in hydrological forecast use case

Branislav Šimo
Ondrej Habala
Viet Tran
Peter Krammer
Ladislav Hluchý
Institute of Informatics of the Slovak Academy of Sciences
Dubravska cesta 9
84507 Bratislava, Slovakia

Abstract—This paper presents successful use of the data mining and integration (DMI) tools created by the EU ADMIRE project in the data mining based hydrological forecast use case. We present general architecture and tools of the ADMIRE framework, DISPEL language for description of distributed workflows, Eclipse based workbench for DMI workflow construction and execution and user portal. Then we describe the hydrological forecast use case, its implementation using the tools of ADMIRE project and share some thoughts on advantages of the framework described.

I. INTRODUCTION

In the world of increasing number and volume of distributed data it becomes important to have access to all those heterogeneous data in a seamless and consistent way, to be able to process it without worrying about the details of physical data format or data set access methods. Also, increasing wealth of data enables creation of a new level of knowledge if the data can be combined and examined in new interesting ways. The goal of the Data Mining and Integration Research for Europe (ADMIRE) [1] project is to provide solution for the above mentioned problems – to make data mining and data integration easier and simpler, allowing researchers to focus more on the data processing and production of new knowledge.

In order to better define its target audience the project categorizes the experts involved in data mining process into three groups: domain experts who are interested in extracting the knowledge hidden in their data, data-analysis experts who are looking for new, more efficient or more reliable algorithms and approaches to knowledge extraction and, finally, data-intensive engineers who build the systems and middleware required for data-intensive computing. With a bit of simplification we could say that data-intensive engineers of ADMIRE project built tools that make work of data-analysis experts more efficient and streamlined and life of domain experts easier.

In the rest of this paper we describe the layers of ADMIRE architecture and the tools created to make data mining easier.

This work is supported by projects ADMIRE FP7-215024, APVV DO7RP-0006-08, DMM VMSP-P-0048-09, Projekt ITMS: 26240220029, VEGA No. 2/0211/09.
local resources (databases, file storage servers, and specialized data processing tools), and the Gateways communicate with each other and stream the processed data. A Gateway is operated by means of a document written in a DISPEL language, based on the concept of connecting streams of data through Processing Elements (PEs). PEs accomplish the actual computational and data integration steps in a DMI process. Each PE corresponds to code that the enactment may activate via some mechanism, such as a web-service invocation or an object’s method invocation. A single enactment may contain multiple instances of a particular PE. The whole ADMIRE framework is intended to operate on large data sets, streaming them through PEs acting as filters. The PEs instantiated in a process are called activities, since most of them are OGSA-DAI [5] activities. Some of the PEs are generic (e.g. an SQL reader), and some must be developed specifically for the application process where they figure.

Figure 2 shows the distributed network of Gateways and also a general application of the architecture to our use case. We have connected several local data providers, provided a DMI process description which streams data from them, cleans, filters and repairs it, and performs data mining on the resulting stream. The results are then provided in an accessible manner to the domain experts who have defined the scenario’s goals.

Project ADMIRE’s framework delivers an integrated approach to enable fluent exploration and exploitation of data. That integration encompasses the full path of actions from accessing the source data to delivering derived results. That path includes gaining permission to access data, extracting selected subsets of data, cleaning and transforming data, performing analyses and preparing results in the form required by their destination. As the multiple sources of data encode and organize information in different ways a significant element is data integration, which includes conversions at both syntactic and semantic levels in order to enable data composition.

B. DISPEL Language

The DISPEL language provides a notation of all DMI requests to a gateway. The description of a DMI process is a composition of existing PEs that actually perform task like querying a data source, transforming tuples in a sequence, merging streams, classifying data and so on. Example of a section of DISPEL is below:

```java
import components
use uk.org.ogsadai.SQLQuery;
use uk.org.ogsadai.TupleToWebRowSetCharArrays;
use eu.admire.Results;
use eu.admire.LinearTrendFilter;

//create an instance of PE SQLQuery
SQLQuery query = new SQLQuery;
String expression = "SELECT ......";
| - expression -| => query.expression;
| - "DbSvpResource" -| => query.resource;

//connect output from query to input of filter
query.data => filter.data;

//create an instance of another PE
LinearTrendFilter filter = new LinearTrendFilter();

//transforming data
TupleToWebRowSetCharArrays toWRS = new TupleToWebRowSetCharArrays;
filter.result => toWRS.data;

//delivery result
Results result = new Results;
| - "results" -| => result.name;
toWRS.result => result.input;
```

The inputs of each processing element may be literal (strings, numbers, variables or expressions of the previous types), or stream outputs from other processing elements. In the example above, we have a SQLQuery activity with two input parameters: SQL command stored in expression variable and a constant string DbSvpResource for data resource. The output from SQL query will be streamed to a filter LinearTrendFilter which replaces missing values. The data then are transformed to a human-readble form and delivered to the result.

It is important to know that data integration and mining is an iterative process, the users can try to use some data sets and mining methods, examine the results, and then change the data sets and/or mining methods to get better results. With DISPEL language, the users can change the data integration and mining process easily.

During execution, the gateways can try to do several optimization techniques to improve performance, e.g. placing processing elements close to data sources. All data are passed between processing elements as streams, i.e. the second processing element can start immediately when the first one produces first data item.
III. TOOLS

The ADMIRE framework provides several tools, which allow users to access ADMIRE services and the applications which use them in a comfortable manner. One of ADMIRE’s initial goals was to provide easy to use and mature data integration and data mining environment to users, both from the data mining domain, and also from the respective application domains. This differentiation among users is the reason that there are several separate user interface options in the ADMIRE framework, namely the ADMIRE Workbench, and the ADMIRE Application Portal. While the tool-rich workbench caters to expert data mining and data-aware distributed computing users, the portal is intended to wrap the results of their work, and to make it accessible to the domain experts, who are not always skilled in data integration or data mining techniques.

A. The ADMIRE Workbench

The ADMIRE Workbench [7] (shown in Figure 3) is a set tools supporting various aspects of DMI work, typically done on the client side by DMI experts. The aim of the Workbench is to support the data understanding, data processing, data integration and modeling stages of the CRISP-DM [8] industry standard methodology for data mining and predictive analytics.

In the ADMIRE project the Workbench interacts with three services:

- Gateway - a gateway to the ADMIRE platform to which the Workbench submits the DMI requests in DISPEL language for enactment.
- Registry - holds the descriptions of all of the components manipulated by the tools on this workbench.
- Repository - a local or remote repository used to store work in progress, such as partially complete DMI-process definitions, or work that a workbench users wish to share more widely or to store more reliably.

The Workbench is a standalone application based on Eclipse [9] application framework, which is written in the Java language. The Eclipse platform provides basic infrastructure and development tools for development of modular Java based applications. An Eclipse based application consists of a set of modules (plug-ins) that implement certain interfaces specified by the platform, which allows the modules to integrate seamlessly into one application. The modules themselves can specify additional interfaces for integration with other modules.

Following modules have been implemented for the Workbench:

- Process Designer - the core of the Workbench. Allows graphical composition of data sources and processing elements into abstract DMI processes.
- DMI Models Visualizer - visualizes data mining models stored in PMML format produced by some of the ADMIRE processing elements.
- Chart Visualizer - visualizes result sets as different kinds of charts, such as pie charts, scatter plots and histograms.
- Gateway Process Manager - keeps track of running Gateway processes and forwards the results from completed processes to other plug-ins such as visualizers.
- Registry View - retrieves lists of available processing elements and types from the ADMIRE Registry, together with their detailed descriptions.
- Semantic Knowledge Sharing Assistant - provides context based help, recommendations and links to resources. The context is detected based on the user interface elements the user is working with or it can be specified by the user
using free text formulation.

B. The ADMIRE Portal

There is no single generic portal in the ADMIRE. Instead a set of application oriented portals have been developed based on the needs of individual applications. All these portals are built using the Google Web Toolkit - the development toolkit for building and optimizing complex browser-based interactive applications using the Java language. They also share the use of the ADMIRE client libraries for components such as Gateway, Registry, Repository, etc.

One of the portals is the Flood Application portal for ORAVA scenario, which aims to provide hydrology experts with domain oriented user interface for researching hydrological data from the Orava region in Slovakia. The graphical user interface (GUI), shown in Figure 4, exploits data mining and integration workflows developed and pre-configured by data mining experts in the context of the ADMIRE project using the Workbench. The portal uses common job execution and management infrastructure provided by the ADMIRE project.

IV. HYDROLOGICAL FORECAST USE CASE

As an example use case to demonstrate usage of ADMIRE tools and infrastructure we have chosen a scenario concerned with predicting water level and water temperature in the river Orava fed from a large reservoir placed in the north of Slovakia. The scenario uses data from several data providers, integrates them and performs data mining on them, providing predictions of the target variables to domain experts (please see [6] for more details about the domain-related properties of this scenario).

A. Technical Description

Figure 2 shows the interconnection of the scenario’s input data, the DMI experts who have designed the DISPEL description of the scenario’s DMI process, and the application domain users who access the data and the process via a custom application-specific portal. The DMI process can be divided into three phases: data integration, training and prediction. The first phase, shown schematically in Figure 5 integrates required data from data sources and saves the result to a file repository in the form of a stream of tuples. The process begins by extracting data about the target hydrological measurement station from a relational database (Station water level, Station discharge, Station water temp). The read values are merged into the initial tuple, which is then further expanded as the process progresses by means of a simple Tuple merge component (employed numerous times throughout the process). Parallel to accessing the waterworks database, we read data from the reservoir database (operated by a separate entity), and also access various parameters present in the computed weather data stored in the form of GRIB files. The data from the reservoir needs to be filtered by the service Orava Reservoir Linear Trend, which replaces missing values in the data by a linear interpolation. Also, any access to a particular GRIB file is preceded by access to the GRIB metadata database,
Fig. 5. A graphical representation of the data integration process in the hydrological scenario of ADMIRE predicting water level and water temperature in a river.

which holds information about the contents of the GRIB files. After integrating all these data into a wide tuple, it is filtered to remove duplicate occurrences of some parameters (for example the date and time, which are used by Tuple Merge to synchronize the data stream), and the result is stored in file storage for later use.

The second phase then reads the integrated data from the file storage, de-serializes it (service Transformation to tuples) and builds a linear regression classifier using parameters set and verified by a data mining expert. The trained model is then serialized and stored back into the file storage.

The final part of this scenario is the prediction phase. This phase expects to find already integrated data in the file storage, created by the integration process (Figure 5). It also downloads the trained data mining model, feeds the integrated data into the model, and then merges the original data with the predicted ones. This process may be used for both verification of the model, if we use past data for prediction, and for the actual prediction, if we use data containing future weather prediction.

B. Lessons Learned

The DISPEL language allowed us to describe the processes of our scenario at a high level of abstraction, independently of any low-level concerns regarding the underlying enactment engines, databases or any consideration of the distributed environment. Our experiments have made use of several interconnected gateways, which together provide all the necessary data, processing elements and visualisation tools which our scenarios require.

This novel approach would enable us to extend easily our data infrastructure to new data providers, by deploying a gateway at the site of the new provider, and registering it with the other gateways. Then, when a data analysis expert creates a DISPEL document that makes use of one of the capabilities provided by this gateway, it can be accessed and integrated automatically into the overall knowledge discovery workflow.

Additionally, this approach allows us to use remote high-performance data mining tools, or to access other data storage facilities.

This model provides a clear separation of responsibilities between data-intensive engineers, data analysis experts, and the domain experts of the application. The underlying infrastructure and gateway network is managed by the data-intensive engineers. The data analysis experts speak DISPEL and create full knowledge discovery workflows which use the infrastructure without needing to understand it; in turn these workflows are used by the domain experts via specialised domain-specific portals.

Apart from separating the concerns of the involved stakeholders, this approach also separates the technology into fairly independent layers, and so a once-tuned DISPEL document will work even when the infrastructure changes significantly, provided that infrastructure is still capable of providing all of the processing elements referenced by the document.

This approach also provides for a reasonable amount of fault tolerance. If one data centre becomes unavailable, it may conceivably be replaced transparently by a different one, without the final users of our product ever knowing it happened. Some centres and gateways are, of course, irreplaceable in a given network (primary data storage centres for instance), but data filters may be deployed at several locations to enhance redundancy. There may be also several HPC facilities available to a given user, so the temporary inaccessibility of one of them is no issue – the DISPEL description of the required data-oriented solution is entirely agnostic of such things.

V. CONCLUSIONS

The ADMIRE architecture, framework and language provide a state-of-the-art DMI platform, and we have shown that it can be successfully applied to a scenario from the environmental domain, performing hydrological predictions which are of interest to actual domain experts. The scenario uses several data sets, which are geographically distributed. The ADMIRE Framework allows us to stream, filter, repair the data, use it in a data mining model, and predict the required target variables. This whole process is conveniently encapsulated in a document written in a high-level, abstract DMI language, also developed in the project ADMIRE, and the process as well as its results can be conveniently accessed from a custom web-based portal, also developed using the ADMIRE technology.
REFERENCES


