Towards Online Data Mining System for Enterprises

Jan Kupčík[1] and Tomáš Hruška[1,2]

[1]*Faculty of Information Technology, Brno University of Technology, Božetěchova 1/2, 612 66 Brno, Czech Republic*

[2]*IT4Innovations Centre of Excellence, Božetěchova 1/2, 612 66 Brno, Czech Republic*

Keywords: Data Mining System, Knowledge Discovery, Data Stream, OLAP.

Abstract: As the amount of generated and stored data in enterprises increases, the significance of fast analyzing of this data rises. This paper introduces data mining system designed for high performance analyses of very large data sets, and presents its principles. The system supports processing of data stored in relational databases and data warehouses as well as processing of data streams, and discovering knowledge from these sources with data mining algorithms. To update the set of installed algorithms the system does not need a restart, so high availability can be achieved. Data analytic tasks are defined in a programming language of the Microsoft .NET platform with libraries provided by the system. Thus, experienced users are not limited by graphical designers and their features and are able to create complex intelligent analytic tasks. For storing and querying results a special storage system is outlined.

1 INTRODUCTION

Nowadays, in our modern, computerized world the vast majority of companies use information technologies to support their activities. Various systems continuously produce, process or store large amount of data that represents messages sent between employees, accounting information, purchases of business clients, log messages from enterprise applications and systems, measurements produced by sensors and much more. Although this data comes from different systems and represents a variety of information, it has one thing in common – it may contain potentially interesting and useful patterns and relationships. The process of revealing new knowledge represented by these discovered patterns from raw data is called data mining or knowledge discovery from databases (KDD) (Han et al., 2011). Understanding relationships that are hidden in the data can contribute to increasing the efficiency of business processes, improving the quality of offered services or discovering new market opportunities and thus increasing company profits.

In the field of data mining a large number of algorithms, techniques and approaches has been created since its beginning in the late 1980s. These new findings have been implemented and formed into many libraries, tools and software suites. In 2011, the overview of existing data mining tools was published (Mikut and Reischl, 2011) – authors identified more than 50 commercial and more than 30 free and open-source software. The lists contain well-known products for corporate usage, research activities and education, for example, IBM SPSS Statistics, MATLAB, Oracle Data Mining, SAS Enterprise Miner and Microsoft SQL Server Analysis Services. From the group of free and open-source tools it can be mentioned Orange, R, RapidMiner or WEKA.

To analyze enterprise data and mine in it, proper software needs to be chosen. One of the most common tasks is to reveal patterns in gathered historical data that is already stored in various types of storages like relational or object-relational databases, OLAP cubes, multimedia documents and other heterogeneous and legacy databases. For that purpose a lot of software exists, but it differs in supported data sources, available mining algorithms, the amount of data it can handle, features related to results visualization, capabilities to export results for further external processing and others. For example, the RapidMiner offers many features, however, it does not support connection to OLAP servers and thus it is not very suitable for repeated mining in these multidimensional sources. Mostly classic mining algorithms, which require multiple passes over the data sets, are present in these tools.

The approach 'store then analyze', which is employed in common data mining algorithms, is not suit-

Kupčík J. and Hruška T..
Towards Online Data Mining System for Enterprises.
DOI: 10.5220/0004098101870192
In *Proceedings of the 7th International Conference on Evaluation of Novel Approaches to Software Engineering* (ENASE-2012), pages 187-192
ISBN: 978-989-8565-13-6

able for all data. Some data that is continuously arriving in potentially high fluctuating data rates like network logs or stock prices must be often analyzed as soon as it emerges and knowledge models must be immediately updated (Chandramouli et al., 2010). In such a case we speak about *data streams* that can be defined as a real-time, continuous, ordered (implicitly by arrival time or explicitly by timestamp) sequence of items (Golab and Özsu, 2003). For data stream mining a number of algorithms and techniques has been proposed (Gaber et al., 2005). From the point of view of software solutions designed for stream processing and analysis, we have identified two groups. The first one is represented by research projects like Stream Mill Miner (Thakkar et al., 2011) or Debellor (Wojnarski, 2008) that focus on data stream mining. In the second group there are so called Data Stream Management Systems (DSMSs) which are designed for receiving data streams from external sources, their processing and pushing results to the output. They primarily enables transforming, filtering and joining data streams from multiple sources, managing sliding windows and processing continuous queries over data streams (Chandramouli et al., 2010). This group includes research projects like STREAM (Arasu et al., 2004) or Borealis (Abadi et al., 2005), as well as commercial products, e.g., Oracle CEP, Microsoft StreamInsight, IBM InfoSphere Streams and many others (Fülöp et al., 2010; Chandramouli et al., 2010; Hébrail, 2008).

We believe that a data mining system for enterprises should be robust, extensible, high performance system which natively supports not only knowledge discovery from both stored data and data streams but also other features like definition of intelligent processes or advanced storage for data mining results. However, to the best of our knowledge, there has been no such a system presented so far. Existing software rather focuses on a subset of these features. In this paper, we briefly introduce our solution that is focused on satisfying all these requirements.

This paper is organized as follows. In the next section the key requirements for our system are described. Section 3 outlines architecture of the system and its components and presents important conceptions. The final section summarizes our work and briefly discuss future plans and challenges.

2 SYSTEM REQUIREMENTS

With respect to the primary purpose of the system we have identified a set of key requirements that our system must fulfill.

Support for Various Data Sources. The system should be able to read and process data from various sources. Support for operating on data stored in relational databases and data warehouses is essential. Next, the system must allow processing of continuous data streams. Thus, the environment will be also prepared for incremental data mining algorithms. Moreover, it should be able to work with other supplementary information that is obtained by transformation of received data, or ask for data from other sources, e.g., web services.

Besides, it is unacceptable to noticeably increase load of data sources related to data querying. The impact to these sources must be minimized.

Advanced Analytic Processes. An analytic process describes what steps the system shall perform to acquire desired results. For example, these steps can represent data loading, data transformations, statistical characteristics computation, execution of mining algorithms, results exploring, storing or further processing. The majority of systems for mining knowledge from data have some kind of a graphic designer to design such processes. They have very broad possibilities to define the data flow, however, means for defining the control flow are very limited, if any. Our system must support intelligent decision-making processes with rich abilities to describe data and control flows. Conditions, cycles, variables and constants are required as well as the possibility to create reusable units, thereby user can perform the same advanced analysis on different data sources without any necessity to intervention.

For processes to be self-reliant one must think about the moment of their execution. The system should still support execution on a user demand, but there must exist a way to start a process as a result of the occurrence of an event. Such an event could emerge when a specified condition is met or a periodic timer rings. This feature guarantees that reports with discovered knowledge may be automatically created.

A Storage for Data Mining Results. A result of a data mining can be a learned knowledge model, revealed patterns and prediction/classification results. It should be possible to store these results to a storage and track history of results. This allows users to view not only recently stored results but also older ones, compare them and study their evolution.

System as a Service. The system must consist of a server and a client part. The server part will run as a service on the server with a sufficient performance for handling awaited load. A client access to the server will be ensured by a web application that offers functions for processes managing, results exploring and system administration. Moreover, the system must be constructed in such a way that another external application can easily connect and consume offered services. Thus, the system should expose a set of services with a well-defined interface available through open multiplatform technologies like web services.

Extensibility. The data mining field is constantly in a progress. New data mining algorithms, techniques and tools are developed and existing ones are improved. To employ new findings from this area, the system must be able to evolve. The aim of our project is to create system whose set of algorithms and tools for visual presentations is easily extensible and upgradable. This system should offer to programmers the API not only for developing new data mining algorithms but also for linking external mining libraries implemented in C/C++.

24/7. One of the most important criteria is suitability for environments where a very high availability and reliability is required. The system can process high number of concurrently running data mining processes whose time necessary for completion may significantly differ and the situation where in every moment a process is running may arise. In many existing data mining systems, if an algorithm should be installed or upgraded, it must be restarted or – in the worst case – recompiled. However, in this system such an approach is unsuitable. All modifications in the set of data mining algorithms must be realizable during system operation. Even if an algorithm is currently running, it must be possible to upgrade it.

3 SYSTEM ARCHITECTURE

The discussed system, whose architecture is presented in Figure 1, is based on a client–server architecture where the server part provides services to many clients. Separation of the client and server parts allows to achieve higher performance, scalability and reliability, because the server part can be operated on a remote high performance machine whereas client applications can run on common personal computers without any special requirements.

The server part consists of several components and layers. In the middle layer there is a component Data collector responsible for receiving, transforming and passing data streams from external environment to other system parts. There is also a subsystem Process manager that manages and executes analytic processes. Access to data is ensured by a service layer Data connector. It offers a unified interface for accessing data stored in the memory and persistent storages like SQL databases and OLAP cubes. Knowledge base manager for analysis results managing is also accessible through this layer. Querying SQL data sources is controlled by the set of intelligent SQL query queues which enable user and system defined priority of queries and ensure optimal loading of database storages. This part is mainly important in environments with limited hardware resources where storages are heavily shared. The objective of the layer called Service providers is enabling to external applications and systems to use services of the mentioned components. For example, an external application for system managing should be able to restart system components or show currently running processes.

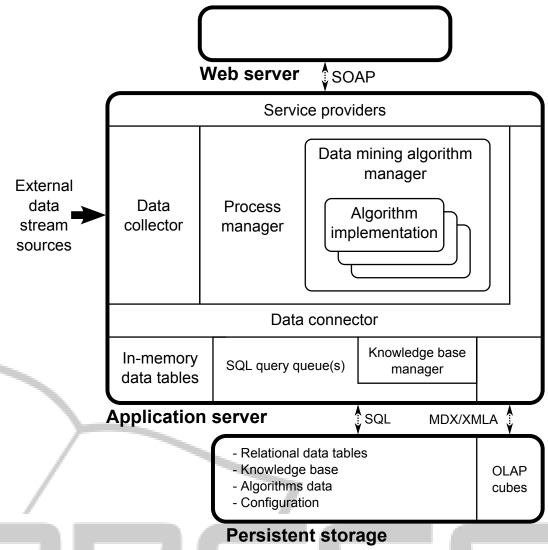

Figure 1: The architecture of the online data mining system.

The server uses services of persistent storages. Data to mine in is read from relational databases and OLAP cubes. Relational databases are also used internally for reading and writing temporary data, configuration and analysis results.

Entire system is being developed on the .NET platform 4.0 in C# programming language. Components of the system are realized as services and communication between them and external systems is ensured by Windows Communication Foundation.

3.1 Analytic Processes

An analytic process defines what actions should the system perform and what data should be employed during the actions. In other words, it represents the data flow and the control flow that lead to desired results.

Processes can be represented in three ways – by an XML-based language, by a special language and by a programming language. For instance, a process definition in a form of an XML document is used in the MSMiner project focusing on cooperating with the OLAP (Shi et al., 2007). As a representative of a special language the Mining Model Definition Language used in the Stream Mill Miner system (Thakkar et al., 2011) can be emphasized. It is also possible to create an analytic process as an self-executable application that uses a library with data mining algorithms, e.g., LIBSVM offering C++ and JAVA implementation of support vector machines (Chang and Lin, 2011).

In most data mining systems complexities of an internal language describing actions to perform are hidden. Users define a process visually by building an oriented graph where nodes are functional components from a predefined set and interconnections between them specify the data flow. The components realize various actions like data loading, transforming, filtering, partitioning, execution of a selected data mining algorithm or showing of results.

Our goal is to provide analysts and process developers as much power and freedom as possible. In our solution, processes are expressed by a common programming language of the .NET platform, for example, by a well-known C# or VB.NET. Thus, experienced users are not limited by visual designers and their features. On the other hand, we realize that not every analyst prefers this level of abstraction of process description. To deal with this situation, we also plan to develop a designer that will enable to define processes visually. Such processes will be automatically convertible to the source code and it depends on users if they need to further edit the source code or directly send it to the system. This tool can also help developers to quickly generate a skeleton of a process.

Each process is represented by a class inherited from the base class common for all processes. This base class exposes an abstract method `Run()` that corresponds to the entry point of the process. The abstract base class is situated in the detachable assembly as well as other classes for data mining algorithm parameters managing, algorithms executing, providing access to the data source services and environment monitoring. To construct a process, two other groups of assemblies are necessary – the first one contains shared interfaces and the second one brings definitions of supported mining models. These assemblies do not have any other references to the system parts and can be easily moved to a process developer's computer.

A developer builds a process with classes from provided assemblies and programming constructs of chosen programming language. Generic classes of the .NET platform can be also involved, however, they must not directly interact with any SQL or OLAP server used by this system. A trivial example of a process can look as follows. An instance of the class representing a data mining algorithm is created, a data mining parameters are added through the `Parameters` property, data sources are defined in a similar way and the algorithm is started by a calling a single method. Then a condition determining if a result should be stored to the knowledge base may be added.

When a process development is finished, it must be deployed to the server. A process may be deployed as a source code that is automatically remotely compiled by the system. The result of compilation is returned to the user and if it is successful, the created assembly and the source code are stored. Later, the source code may be downloaded, modified and deployed again. It is also possible to deploy a process assembly that was compiled with the detachable assemblies. However, in this case the source code is not tracked by the system.

Once a process assembly is obtained, the process manager reads the settings of process activation from the process metadata. Based on the type of activation, the process is placed into a priority process queue ordered by time of a process start or appended to a list of processes activated by the data collector. If the process shall be executed immediately, it is inserted to the beginning of the process queue. The process can be also scheduled for periodic starting, so it is inserted into the process queue to the correct position. The process may be also executed if a condition related to an input data is satisfied. In this case, the condition and the process to be executed is registered in the data collector. As soon as the process manager detects that a process prepared to execution is ready in the process queue or the data collector marks a process as runnable, the entry point of the process is started.

Because a process code is directly executed, a severe bug in it might cause a crash of the process manager and other processes. To deal with this problem, each process is loaded into its own application domain and executed in it. Thus, a domain can be torn down without any impact to other parts of the system. This solution also allows to kill processes in an infinite loop.

3.2 Data Sources

For performing data analysis and mining, data sources need to be defined. There exist two cases depending on the character of data. In the first one, data to process is already stored in some relational databases or OLAP cubes and can be obtained by a SQL or MDX/XMLA query. In this case the system will connect to requested data source, cache data if requested and execute a chosen algorithm for data mining in large data sets.

The second case corresponds to the situation where data is continuously generated by the environment outside the system boundaries and it is not suitable to store them all before mining. To be able to process data streams, the data collector component must be employed. An external system produces data records consisting of an identifier characterizing carried information and data itself and passes them to this component. For each unique identifier the system administrators can set a series of actions that describes transformations and filtering over the data. Each action is tied up with any number of processes which will be executed or notified with output data as a parameter as soon as all steps in the action are finished.

The data collector enables only simple operations over the data. It also supports sliding windows that can hold a fixed number of elements or elements from a time period with certain duration. However, the need for more advanced data stream processing can arise. In such a situation the data collector can be simply extended to cooperate with a Data Stream Management System.

3.3 Knowledge Base

In this system there is a detachable subsystem for storing and managing analysis results called the knowledge base. It consists of a controller labeled as knowledge base manager in Figure 1 and a relational database in which knowledge base data is stored. The controller is the only component that can directly access knowledge base database, the other parts of the system uses services of the controller to write and query information.

The knowledge base storage can preserve various results of data analyses which may be in form of learned knowledge models and sets of revealed patterns obtained by data mining algorithms. It can also store smaller data sets including their schema, so it is possible to hold statistics about source data, for example, sum of all purchases in selected countries in a time period. Each result, which can be referred to as knowledge, is placed into a container within the knowledge base. A container can contain other containers, so hierarchical structures of knowledge may exist.

This subsystem can attach metadata to new knowledge. Currently, there are supported two types of such additional information. In case the knowledge represents the result of a data mining algorithm, the unique name of the algorithm and current values of all its parameters are automatically added. The second type of supported information is a time mark and a time period that can express a range of analyzed data. This time information is optional and it is stored only on user request.

Retrieval of stored entities can be realized in several ways. The API of the knowledge base enables accessing all stored data structures within requested knowledge, their filtering by algorithms, parameters or validity time. For instance, this feature can be helpful to find best settings of the parameters of a selected mining algorithm. It can also simplify the analysis of input data evolution – it may be used in case of market basket analysis with association rules mining where the evolution model can reveal fluctuations in customer preferences.

4 CONCLUSIONS AND FUTURE WORK

There are many software solutions related to knowledge discovery process. They vary in features, capabilities, supported data sources and other aspects. We have explored the set of existing products and found out an uncovered space between them. Our system, which was introduced in this paper, fills this gap by offering a combination of features packaged in one software that is missing so far.

The presented system is designed for analyses of great amounts of data which may be stored in a relational or multidimensional storage or may come on sample-by-sample basis as data streams. The set of data mining algorithms in the system is not fixed, it can be modified or extended, even at run-time. The combination of these features enables to employ this system in time demanding analysis tasks. Results of tasks may be stored into a detachable knowledge base that is able to track their evolution in time. Last but not least important attribute of this system is the way how the analysis processes are defined. They are expressed by a code written in a programming language of the .NET platform in conjunction with a set of libraries provided by the system. All these features were outlined in this paper.

Although this system is functional, there is a lot of space for improvements. The system supports reading from relational databases, OLAP cubes and in-memory tables, but does not enable to combine them freely on the process definition level. The aim will be to allow it to users. The second improvement area relates to the way of processing data. Parts of analytic process definitions might be transferable to database servers that would be able to perform some actions on data without necessity of immediate transfer to the central server. An attention can be also paid to improvements of the knowledge base to increase its querying capabilities.

The system will be also thoroughly evaluated in terms of performance and compared to selected data mining systems. At this moment, preliminary tests indicate that presented approaches are correct. For instance, they have showed the system is capable of obtaining data mining results in shorter time than Microsoft SQL Server 2008 R2 Analysis Services. Other tests will be focused on capability to cope with failures of processes and algorithms that might cause a system crash or affect the system availability.

ACKNOWLEDGEMENTS

This work has been supported by the research programme TAČR TA01010858, BUT FIT grant FIT-S-11-2, by the research plan MSM 0021630528 and the European Regional Development Fund in the IT4Innovations Centre of Excellence project (CZ.1.05/1.1.00/02.0070).

REFERENCES

Abadi, D. J., Ahmad, Y., Balazinska, M., Cherniack, M., Hwang, J.-h., Lindner, W., Maskey, A. S., Rasin, E., Ryvkina, E., Tatbul, N., Xing, Y., and Zdonik, S. (2005). The design of the borealis stream processing engine. In *In CIDR*, pages 277–289.

Arasu, A., Babcock, B., Babu, S., Cieslewicz, J., Datar, M., Ito, K., Motwani, R., Srivastava, U., and Widom, J. (2004). STREAM: The stanford data stream management system. Technical Report 2004-20, Stanford InfoLab.

Chandramouli, B., Ali, M., Goldstein, J., Sezgin, B., and Raman, B. S. (2010). Data stream management systems for computational finance. *Computer*, 43:45–52.

Chang, C.-C. and Lin, C.-J. (2011). LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27.

Fülöp, L. J., Tóth, G., Rácz, R., Pánczl, J., Gergely, T., Beszédes, A., and Farkas, L. (2010). Survey on complex event processing and predictive analytics. Technical report, University of Szeged, Department of Software Engineering.

Gaber, M. M., Zaslavsky, A., and Krishnaswamy, S. (2005). Mining data streams: A review. *SIGMOD Rec.*, 34:18–26.

Golab, L. and Özsu, M. T. (2003). Issues in data stream management. *SIGMOD Rec.*, 32:5–14.

Han, J., Kamber, M., and Pei, J. (2011). *Data Mining: Concepts and Techniques*. Morgan Kaufmann Publishers Inc., Waltham, MA, USA, third edition.

Hébrail, G. (2008). In Fogelman-Souli, F., Perrotta, D., Piskorski, J., and Steinberger, R., editors, *Mining Massive Data Sets for Security, Advances in Data Mining, Search, Social Networks and Text Mining, and their Applications to Security*, volume 19, chapter Data stream management and mining, pages 89–102. IOS Press.

Mikut, R. and Reischl, M. (2011). Data mining tools. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 1(5):431–443.

Shi, Z., Huang, Y., He, Q., Xu, L., Liu, S., Qin, L., Jia, Z., Li, J., Huang, H., and Zhao, L. (2007). MSMiner—a developing platform for OLAP. *Decis. Support Syst.*, 42:2016–2028.

Thakkar, H., Laptev, N., Mousavi, H., Mozafari, B., Russo, V., and Zaniolo, C. (2011). SMM: A data stream management system for knowledge discovery. In Abiteboul, S., Böhm, K., Koch, C., and Tan, K.-L., editors, *Proceedings of the 27th International Conference on Data Engineering, ICDE 2011, April 11-16, 2011, Hannover, Germany*, pages 757–768. IEEE Computer Society.

Wojnarski, M. (2008). Transactions on rough sets IX. chapter Debellor: A Data Mining Platform with Stream Architecture, pages 405–427. Springer-Verlag, Berlin, Heidelberg.