Open Mobile Miner: A Toolkit for Mobile Data Stream Mining

Shonali Krishnaswamy, Mohamed Medhat Gaber, Marian Harbach, Christian Hugues, Abhijat Sinha, Brett Gillick, Pari Delir Haghighi, and Arkady Zaslavsky
Centre for Distributed Systems and Software Engineering
Monash University, Australia
{Shonali.Krishnaswamy, Mohamed.Gaber, Marian.Harbach, Christian.Hugues, Abhijat.Sinha, Brett.Gillick, Pari.DelirHaghighi, and Arkady.Zaslavsky}@infotech.monash.edu.au

ABSTRACT
There is an emerging focus on real-time data stream analysis on mobile/ubiquitous devices. A wide range of data stream processing applications are targeted to run on mobile handheld devices with limited computational capabilities such as patient monitoring, driver monitoring, providing real-time analysis and visualization for emergency calls, optimization of logistics for courier pick-up and delivery etc. In this paper, we present the first generic toolkit for mobile data mining. The Open Mobile Miner (OMM) toolkit is easy to use, can be deployed on a range of mobile devices, is extensible and can be customized for application specific needs. A video of the system in operation for three different settings is available at: http://www.csse.monash.edu.au/~shonali/OMM/OMM-VideoDemo.asf and can be viewed using Windows Media Player™.

Keywords
Data stream mining, Pervasive environments, Ubiquitous computing applications, Sensor data, Adaptation, Resource-awareness.

1. INTRODUCTION
The phenomenal growth of mobile devices coupled with their ever-increasing computational capacity presents an exciting new opportunity for real-time, intelligent data analysis in pervasive/ubiquitous environments. Ubiquitous Data Mining is the process of analyzing data streams using mobile and/or embedded devices (e.g. sensors) to support critical applications such as mobile healthcare, intelligent transportation systems, and emergency/disaster management like bushfires [2]. The typical constraints that have to be addressed in performing mobile data mining are:

- **Data Streams** are generated and sent in real-time in a stream format [17] with little or no potential for persistent storage.
- **Resource Constraints** include limited computational resources such as memory, processor speed, network bandwidth, battery power, and screen real-estate.
- **Temporal Constraints** refer to real-time information and decision-making needs that, in turn, necessitate the analysis to be online, incremental, continuous.
- **Mobility** of users and devices and the connectivity issues thereof.
- **Adaptation** of the analysis process to varying/dynamically changing resource-levels and user needs.

In the last few years, rapid strides have been made in accurately and efficiently mining high speed data streams in mobile devices such as Personal Digital Assistants (PDAs) and there is a growing focus on "in-network" processing using embedded devices such as sensor nodes. These techniques leverage the body of work that exists in mining data streams and aim to enable the operation of these algorithms in resource-constrained environments [2, 6].

In this paper, we demonstrate the first mobile data mining toolkit: Open Mobile Miner (OMM). The primary motivations for the development of this toolkit are as follows:

1. Enable easy deployment of mobile data mining applications on a range of mobile devices;
2. Provide a platform for evaluation of new and existing mobile data stream mining techniques by the research community;
3. Encapsulate extensibility of the toolkit by easy addition of new capabilities;
4. Facilitation integration of new and existing data stream mining algorithms into the toolkit that may or may not have adaptation mechanisms incorporated;
5. Interface with a range of input sources for data streams including Bluetooth-enabled sensors, previously recorded data, distributed data, and synthetic data (i.e. data stream generation for evaluation purposes);
6. Allow flexible, application specific visualizations to be developed.

Thus, the above considerations also form the requisite functionality that has driven the development of the OMM.

The rest of the paper is organized as follows: Section 2 presents the conceptual ar

chitecture of the Open Mobile Miner (OMM) with a discussion on the range of sensory data and synthetic data it can access, the algorithms that have been implemented, and the visualization

KDD'09, June 28–July 1, 2009, Paris, France.

strategies. Section 3 presents the implementation and operation of the Open Mobile Miner. Section 4 presents the applications/demonstrations of the toolkit in the real-life case studies. Finally, the paper is concluded in Section 5.

2. Conceptual Architecture of OMM

Having presented an overview of the adaptation process and its functioning, we now discuss the conceptual architecture of the Open Mobile Miner, shown in Figure 1 below.

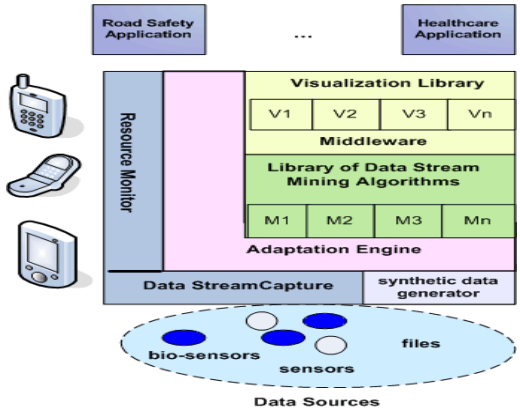

Figure 1. The architecture of Open Mobile Miner (OMM)

The key components of the architecture are as follows:

Data Sources: The streams of data that need to be analyzed are generated at the data sources. Data sources include sensors which perform monitoring/measurements and transmit this data continuously such as environmental sensors that measure physical phenomena (e.g. temperature, pressure, soil-humidity), or bio-sensors that measure physiological phenomena (e.g., heart-rate, ECG, movement levels etc.). These sensors can communicate/transmit their data via various communication channels (e.g. Berkely MOTES which can measure light, sound, and movement send data via 802.11 while Alive Tech ECG sensors transmit via Bluetooth). In addition to being able to receive live sensory data, a toolkit for Mobile Data Mining must also be able to support for testing/evaluation purposes previously recorded data that can be re-played as a stream (simulating sensory input) as well as be able to generate synthetic data streams according to various distributions required. Thus, the Open Mobile Miner can receive data from four different sources:

- sensors that transmit either though Bluetooth or WiFi;
- a data generator that can generate a specified number of streams each with a specified distribution (e.g. Binomial, Gaussian, Poisson, Uniform etc.), for the specified parameters;
- read recorded in a local CSV file and re-play it as stream;
- replay the contents of a CSV file as stream from a web source (i.e provide a basis for simulating a distributed/remote data stream).

Data Stream Capture: This component receives data streams from the various sources and passes it either to the data stream mining algorithms or the adaptation engine depending on whether analysis process has been initialized to operate in adaptive manner or not. This component may perform some buffering of data so as to enable determining the data rate and preventing loss of data.

Library of Data Stream Mining Algorithms: This is the analyzer library which provides a range of data stream mining analysis algorithms for mobile data mining. These algorithms perform varied types of analysis including: clustering, classification, frequent items, change detection and time series analysis. The algorithms can either operate by leveraging the principals of adaptation as discussed above or can operate without adaptation to cater for techniques that may not have been built with adaptation strategies. When an algorithm is operating in adaptive mode, it liases with the Adaptation Engine in processing the input streams. However, when the algorithms are functioning in non-adaptive mode the incoming data streams need to be processed directly by the algorithms and the Adaptation Engine and Resource Monitors discussed below remain inert/inactive. Table 1 shows the implemented algorithms in OMM.

Table 1 OMM Algorithms

Type	Algorithm
Clustering	LightWeight Cluster (LWC)
	RA-Cluster
	RA-VFKM
Classification	LightWeight Classification (LWClass)
Change detection	Change-Detect
Time series analysis	RA-SAX
Frequent pattern	LightWeight Frequent items

Adaptation Engine: This component manages the adaptation process in terms of obtaining information regarding the data stream characteristics (e.g. data rates) from the data sources as well as resource-levels (i.e. status of computational resources including battery levels) of the device and instrumenting the performance of the data stream mining algorithms according to this information. The Adaptation Engine has built-in strategies for adjusting dynamically the functioning of the data stream mining algorithms according to the various parameters by varying accuracy levels. This is achieved according to the principles outlined previously in Section 2 and includes dynamically performing knowledge integration, or reducing frequency at which the data stream is processed. For further details about the adaptation of the other algorithms implemented in OMM, readers are referred to [3,4,5,7,8] for a thorough discussion and formalization of the different strategies used in adjusting the algorithm parameters according to resource consumption.

Resource Monitor: This component is responsible for assessing the levels of memory, processor and battery that is currently available on the device and in conjunction with the data stream rates constitute the principal basis for performing adaptation. This component primarily communicates the resource level information to the Adaptation Engine. This component is – unlike the others – operating system specific. Given the range of mobile devices that are being developed and their diverse operating systems (e.g. Nokia phones run the Symbian OS, Google GPhone runs the Android OS and the iPhone runs iPhone OS from Apple) – this component has to implement the OS specific functions to access low-level computational characteristics.

Visualization Library: The visualization library allows the results of the analysis process to be shown using custom visualization techniques. Given that many applications will require custom visualizations, the framework needs to facilitate integration of

application specific visualization. The visualization middleware performs the task of obtaining the output of the algorithms (e.g. cluster details) as they are available and also maintains information regarding visualization preferences (e.g. colors and shapes used to represent clusters). This information needs to be provided while initiating/configuring the analysis task. It is noteworthy that visualization of data stream mining on mobile devices is very much an emerging area of study. In this context, this component at this stage provides simple visualizations only. The challenges involve coping with incremental results, dynamic changes in the analysis results and coping with the limited screen real-estate that needs to manage screen-clutter as it evolves.

3. IMPLEMENTATION AND USAGE OF THE OMM TOOLKIT

The motivation for the development of the Open Mobile Miner (OMM) was to provide a generic tool to facilitate research on mobile data mining. The key underlying characteristics that have driven the implementation are: ease of use, flexibility in dealing with a range of mobile devices and varied data, customizability in terms of easily including application-specific visualizations/output mechanisms and extensibility in terms of being able to include new data sources such as new sensors, analysis algorithms, adaptation strategies and visualization techniques. The extensibility is driven by the engineering of the toolkit and provision of specific interfaces that allow new components/features to be easily added. The OMM toolkit is split into two parts: A Core that provides all the functionality needed to do adaptive mobile mining and a graphical user interface (GUI) that facilitates ease of use for the Core's functionality through graphical controls. Figure 2 illustrates how OMM works when invoked from a GUI and how the core components interface.

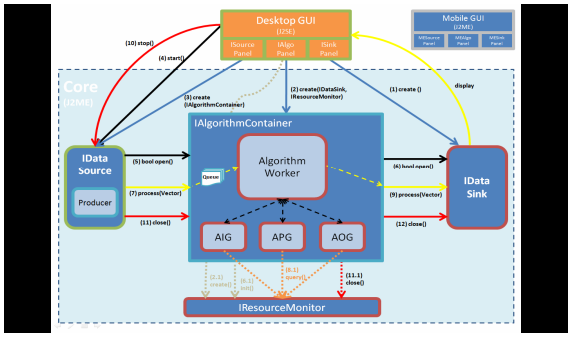

Figure 2. Implementation Structure of the OMM Framework

For portability reasons, the Core had to be entirely implemented in Java ME. Hence it runs on all recent Java Runtime Environments (JREs), be it on a mobile handset, a PDA or a Java EE application server. Both parts, the GUI and the Core, are packaged separately. At present two different GUI implementations exist: one using Swing and Java SE and another one using LWUIT[1] and Java ME. Both make most of the core's functions accessible via graphical controls.

The basic idea is a push-based pipeline for data to be analyzed. A *Data Source* acquires or generates data elements, pushes them one by one into an *Algorithm Container* which in turn may output results to a *Data Sink* whenever necessary. This behavior is shown as the yellow arrows in Figure 7. Within OMM's core, the data just keeps flowing upstream through an algorithm. The data source acts as an adaptor for the system to the incoming data stream converting items into the necessary format. In turn, the data sink can be used to transform results into any desired format for visualization. OMM's GUIs aim at providing a way of setting up the data path using graphical controls for convenience and experimental purposes. The core functionality is accessible from the GUI by selecting the components to connect. The user is required to enter the necessary parameters for the respective source, sink or algorithm and can eventually run the system. Furthermore, a tight integration with any software can be achieved by accessing the OMM Core functions directly via the API. This is done in a straightforward manner by instantiating component classes directly. Figure 3 shows a screenshot of the OMM Desktop GUI. To setup the system, one selects source, algorithm and sink, as shown in steps 1 to 3. After pressing the select button, a tree of available components is shown. After making a selection, a box containing the available parameters is displayed allowing adjustment of the component's behavior as required.

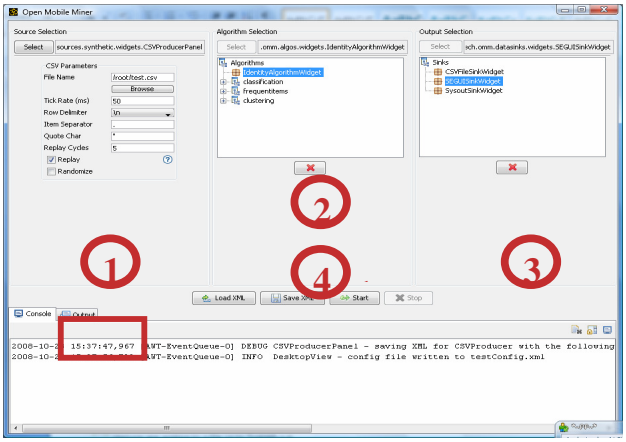

Figure 3. How to use the OMM Desktop GUI

The OMM GUI will pick this up and display it as an option in the respective component's tree listing. After the component choices are made, the system can be run by hitting the start button (shown as Step 4). The execution can be stopped by hitting the Stop button at any time. Another option is to save the current selection and configurations from the widgets into an XML file. This file can then be loaded back into the GUI at another point of time or deployed on a mobile device and used to run OMM without having to configure it manually beforehand.

The mobile GUI is similarly structured to the Desktop GUI. It can be configured to load configurations from an XML file previously generated by the Desktop GUI using the "Load" option on the welcome screen. If the configuration file is correct and complete the "Run" option will appear in the lower right corner. This is shown in Figure 4. Alternatively, it can be setup manually using the same steps shown in Figure 3.

The main components of OMM Core are three main interfaces: *IDataSource*, *IAlgorithmContainer* and *IDataSink*. Furthermore, two utility interfaces provide support for resource awareness and runtime statistics: *IResourceMonitor* and *IStatsConsumer*. The currently available sources include:

[1] https://lwuit.dev.java.net/

AliveTechHM131BTProducer: This uses Bluetooth to interface with an AliveTech ECG Heart Monitor™ device and parses relevant data from the provided data stream.

CounterProducer: A producer mainly used for debugging purposes. It generates a stream of increasing Integers, with configurable dimensionality, start, stop and step values.

CSVProducer: This producer reads comma separated values from a CSV file and passes each line as one vector into the algorithm. It can replay the file unboundedly if necessary.

RandomProducer: This producer uses the uncommons-maths[2] Random Number Generators (RNGs) and probability distribution wrappers to generate Vectors of random numbers that fit different distributions. It has been ported to Java ME, but direct use on the mobile device is discouraged, since number generation is computationally expensive. All Producers offer a parameter to configure the rate with which data is produced. This is basically the time the Thread waits after one element was handed to the algorithm. OMM Core provides resource monitor implementations for the following platforms: **J2SEResourceMonitor** for Windows XP/Vista, Windows CE and a so-library for Linux.

S60ResourceMonitor: For use on Nokia S60 devices, a special Resource Monitor has been provided.

SimulatedResourceMonitor: The *SimulatedResourceMonitor* can be used for testing purposes or if no other resource monitor is applicable. It uses random values for CPU and memory usage and simulates a linear degradation of battery power.

Figure 4. Loading an XML Config File in the Mobile GUI

4. DEMONSTRATION

OMM is currently being applied in two areas. We are currently working with the Centre for Accident Research and Road Safety – Queensland (CARRS-Q) in applying OMM and our adaptive UDM algorithms in the area of Intelligent Transportation Systems [9]. The second application is in the area of cardiac monitoring. This is a collaborative project involving cardiologists from the Dept. of Cardiovascular Research at Monash University and the Alfred Hospital in Melbourne, Australia and the Dept. of Biomedical Engineering at RMIT University, Australia. The system that is being built for monitoring patients involves using bio-sensors (both commercial including the Alive Tech Heart Monitor™ as well as those built in-house at the Dept. of Bio-Medical Engineering, RMIT), obtaining other parameters such as body-weight. This project relies on situation-aware fuzzy rules along with change detection [4] to see monitor heart rate changes.

[2] https://uncommons-maths.dev.java.net

Some custom visualisations built for this application are shown in figure 5.

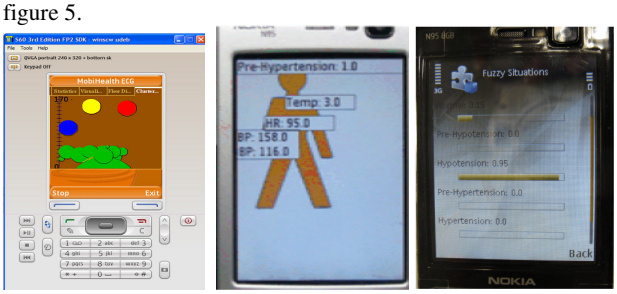

Figure 5 Heart Rate Analysis Using OMM

5. CONCLUSIONS

This paper has presented to the academic and practitioner data mining community an important new resource – a mobile data mining toolkit. We have outlined the theory of adaptation for performing mobile data mining, presented in detail both the conceptual framework and implementation of the Open Mobile Miner. The OMM toolkit has been built such that it is easy to use, extensible by the inclusion of new data sources, analyzers, adaptation strategies, visualizers and customized for specific applications. Furthermore, it can also be tailored to operate on a range of mobile devices by merely including a Resource Monitor for that particular OS.

6. REFERENCES

[1] Domingos, P. and Hulten, G. 2001. A General Method for Scaling Up Machine Learning Algorithms and Its Applications to Clustering. Proceedings of the 18th Int. Conf. on Machine Learning,.

[2] Gaber, M, M., Krishnaswamy, S., and Zaslavsky, A. 2005. On-board Mining of Data Streams in Sensor Networks, A Book Chapter in Advanced Methods of Knowledge Discovery from Complex Data, (Eds.) S. Badhyopadhyay, U. Maulik, L. Holder and D. Cook, Springer.

[3] Gillick B., Krishnaswamy S., Gaber M. M. and Zaslavsky A. 2006. Visualisation of Fuzzy Classification of Data Elements in Ubiquitous Data Stream Mining. IWUC 2006, 29-38.

[4] M. M.Gaber, P. S. Yu, "Detection and Classification of Changes in Evolving Data Streams", International Journal of Information Technology & Decision Making, Vol. 5, No. 4, World Scientific Publishing Company, 2006.

[5] Gaber, M, M., Zaslavsky, A. and Krishnaswamy, S. 2004. A Cost-Efficient Model for Ubiquitous Data Stream Mining. Proceedings of the 10TH Int. Conf. on Information Processing and Management of Uncertainty in Knowledge-Based Systems, Perugia Italy, July 4-9.

[6] M. M. Gaber, A. Zaslavsky, S. Krishnaswamy, "Mining Data Streams: A Review", ACM SIGMOD Record, 34, 1 (June 2005).

[7] Shah R., Krishnaswamy S., and Gaber M. M. 2005. Resource-Aware Very Fast K-Means for Ubiquitous Data Stream Mining. Proceedings of 2nd Int. Wshop on KD in Data Streams, ECML/PKDD 2005.

[8] Gaber M. M., Yu P.S., A Holistic Approach for Resource-aware Adaptive Data Stream Mining, Journal of New Generation Comput. 25(1) pp. 95-115, 2006.

[9] Salim, F. D., Loke, S. W., Rakotonirainy, A., Srinivasan, B., Krishnaswamy, S., 2007, Collision pattern modeling and real-time collision detection at road intersections, Proc of the 2007 IEEE Intelligent Transportation Systems Conference, USA, pp. 16