Multivariate Stream Data Classification Using Simple Text Classifiers

Sungbo Seo[1,*], Jaewoo Kang[2,3], Dongwon Lee[4], and Keun Ho Ryu[1]

[1] Dept. of Computer Science, Chungbuk National University, Chungbuk, Korea
{sbseo, khryu}@dblab.chungbuk.ac.kr
[2] Dept. of Computer Science and Engineering, Korea University, Seoul, Korea
[3] Dept. of Computer Science, North Carolina State University, Raleigh, NC, USA
kang@csc.ncsu.edu
[4] College of Information Sciences and Technology, Penn State University, PA, USA
dongwon@psu.edu

Abstract. We introduce a classification framework for continuous multivariate stream data. The proposed approach works in two steps. In the preprocessing step, it takes as input a sliding window of multivariate stream data and discretizes the data in the window into a string of symbols that characterize the signal changes. In the classification step, it uses a simple text classification algorithm to classify the discretized data in the window. We evaluated both supervised and unsupervised classification algorithms. For supervised, we tested Naïve Bayes Model and SVM, and for unsupervised, we tested Jaccard, TFIDF, Jaro and JaroWinkler. In our experiments, SVM and TFIDF outperformed the other classification methods. In particular, we observed that classification accuracy is improved when the correlation of attributes is also considered along with the n-gram tokens of symbols.

1 Introduction

Different sensor network applications generate different types of data and have different requirements for data processing (e.g., long term monitoring of sea level change vs. real-time intrusion detection). Different data processing strategies need to be considered for the different types of applications. Even in the same application, the characteristics of data generated in the network sometimes changes over time. For example, in a network monitoring application, users may want to receive only 5% samples of original data when network operates normally, while they might want to receive full data for further analysis when an interesting pattern (e.g., similar to a predefined intrusion pattern) is detected. The ability of handling sensor data adaptively by detecting changing characteristics of data becomes important in many data-centric sensor applications.

In order to address this problem, we propose a scalable framework for multivariate stream data classification that allows using simple, well-understood text classifiers to classify multivariate streams, instead of building custom classification algorithms for

[*] Work performed while the author visited North Carolina State University.

S. Bressan, J. Küng, and R. Wagner (Eds.): DEXA 2006, LNCS 4080, pp. 420 – 429, 2006.

different sensor applications. The proposed method works in two steps as follows. It first discretizes the stream data into a string of symbols that characterize the signal changes, and then applies classification algorithms to classify the transformed data. This transformation simplifies the classification task significantly.

The classification model is learned from a user-labeled data. Users assign a descriptive label to each window in the training set. For example, if the sensor data in a window contains an intrusion pattern, the user labels the window as "intrusion". Similarly, if a window contains normal signals, it can be labeled as "normal". Once the classification model is built, the classifier can start taking new windows of data and predict the labels for the windows. For the classification step, we evaluated both supervised and unsupervised methods. For supervised, we tested Naïve Bayes Model and SVM, and for unsupervised, we tested Jaccard, TFIDF, Jaro and JaroWinkler.

We identify the contributions of our work as follows:

1. In order for fast pattern matching, we discretized the continuous sensor streams into a string of symbols characterizing signal changes. In order to allow partial matches and to retain temporal locality of patterns, we chunked the symbol strings into various lengths of n-grams. This representation gives rise to a large number of widely used string and vector-space classifiers.
2. The proposed framework and the classification model can be utilized for the sensor network querying and monitoring in general. It enables the real-time monitoring of continuous sensor data. Moreover, it can also be used for the analysis of historical data accumulated in a server. Using the method, we can serve ad-hoc queries such as finding windows that have similar data to the input pattern.

The rest of the paper is organized as follows. Section 2 introduces related work. Section 3 describes our multivariate stream data classification methods. Section 4 presents experimental results and Section 5 presents concluding remarks.

2 Related Work

In our problem context, sensor data is an unbounded multivariate time series data. Multivariate time series data classification methods were studied in [4, 5, 6, 7, 8], including On-demand Classifier [4], HMM (Hidden Markov Models) [5], RNN (Recurrent Neural Network), Dynamic Time Warping [5], weighted ensemble classifier [6] and SAX [7]. These methods involve large numbers of parameters and complex preprocessing step that need to be tuned. Due to the dynamic nature of sensor network environment and the diverse types of applications, the applicability and effectiveness of these specialized solutions is not immediately clear for the sensor network applications.

On the other hand, there exist many popular general purpose classifiers that work for string and vector-space models, including Bayesian classifiers [11], Support Vector Machines (SVM) [12] and string-distance based methods [14]. In our proposed approach, we discretize the multivariate continuous time series data into a series of symbols. This transformation allows sensor data to be viewed as a sequence of words consisting of the symbols, giving rise to such general purpose classifiers.

3 Multivariate Stream Data Classification

3.1 Problem Definition

In a hierarchically organized sensor network as shown in Fig. 1, a sensor node represents a collection of heterogeneous sensors collocated in the same geographical location. Each of these sensors monitors or detects different target objects or events. The sensor data generated from such a sensor node collectively forms a multivariate data stream. Each sensor node temporarily accumulates the sensor data and periodically sends it to the parent node in the upper layer. The parent node then collects data transmitted from children nodes and either relays it up to the chain (e.g., from sensor node to base station), or stores them in the repository or feeds them to the application for further processing (server node).

The problem we attempt to address in this paper can be formulated as follows. As illustrated in Fig.1, let $t_i = [s_{1i}, s_{2i}, \ldots, s_{mi}]$ be an m-dimensional tuple, representing sensor readings from *m* different sensors (s_1 to s_m) at time point *i*. Let $W_j = [t_{j*p+1}, t_{j*p+2}, \ldots, t_{(j+1)*p}]$ be a *j*-th window of size *p,* containing *p* tuples from t_{j*p+1} to $t_{(j+1)*p}$. Finally, let $T = [W_1, W_2, \ldots, W_\infty]$ be an unbounded stream of windows. Suppose the

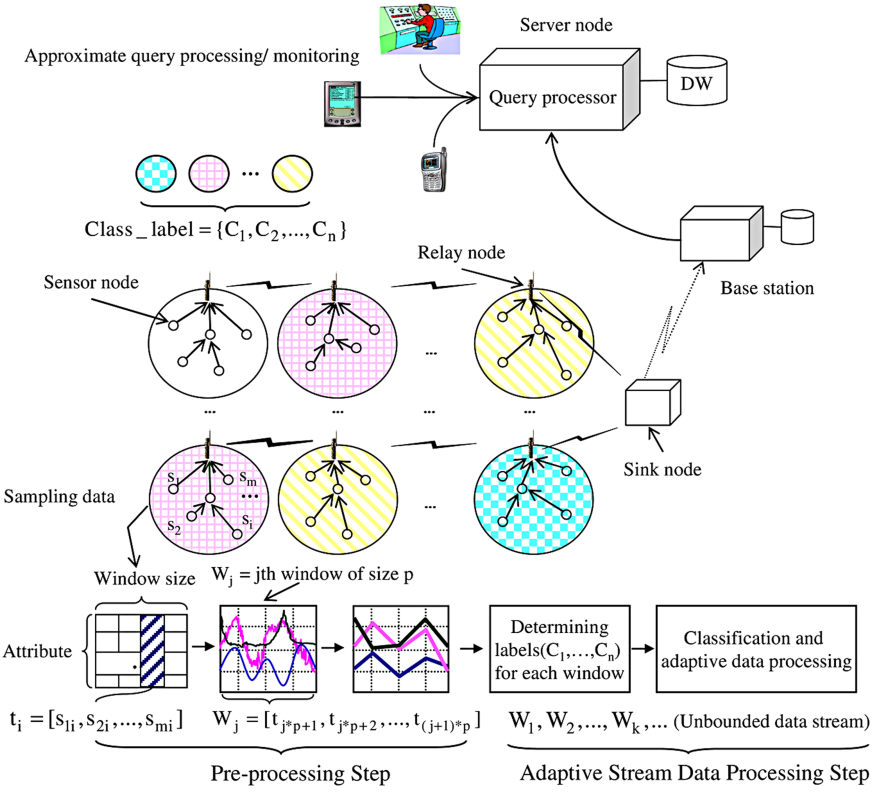


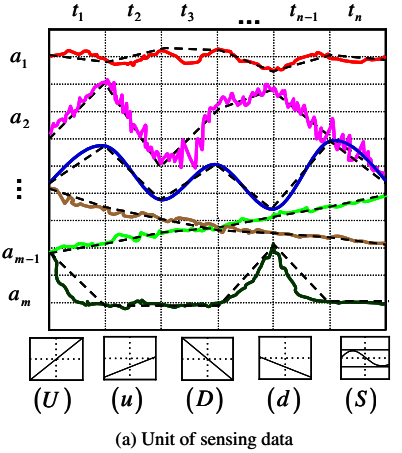
Fig. 1. Overview of real-time data analysis in wireless sensor networks

first k windows (W_1 to W_k) are pre-labeled by the user. Each user labeled window has a class label chosen from *n* labels, C_1 to C_n. Then, the problem is to build a classifier to predict the labels for all subsequent windows (W_{k+1} to W_∞) based on the labeled windows.

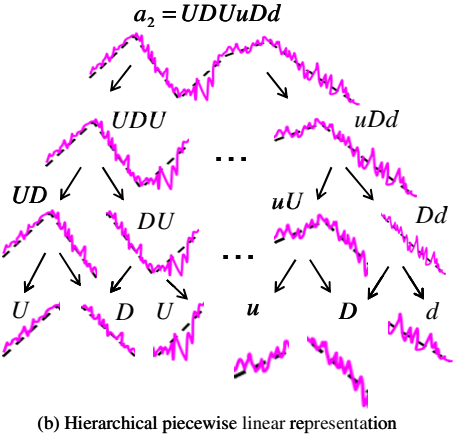
3.2 Preprocessing Step

Fig. 2 shows the preprocessing step for our approach. In this step, the continuous sensor stream is transformed into the combinations of discrete symbols which represent signal changes in each sensor stream, such as upward (*U* for steep inclination and *u* for moderate inclination), downward (*D* for deep and *d* for moderate) or stable (*S*) for a given time interval [t_i, t_{i+k}] where *k* being a constant between 1 and the window size *p*. This transformation greatly reduces the complexity of the raw data while retaining the structure of the time series data. For fast trend analysis and pattern matching, we use a hierarchical piecewise linear representation [9] and n-gram model [11] which together can represent various different types of multivariate stream data. In this paper, we used the five symbols (*U, u, D, d,* and *S*) as shown in Fig. 2(a). All the attributes in a window can be represented as in Fig. 2(b) using this representation.

We extended the original hierarchical piecewise linear representation, which splits the original patterns into a set of disjoint sub-patterns, with n-gram based pattern chunking in order to support partial matches and to preserve the orderings between the



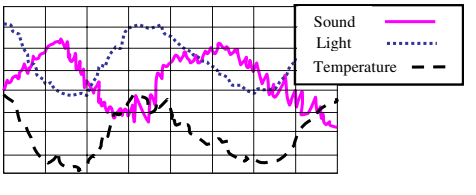
(a) Unit of sensing data



(b) Hierarchical piecewise linear representation

1-gram	*U, u, D, d, S*
2-gram	*UU, Uu, UD, … ,DS, dS, SS*
3-gram	*UUU, UuD , … , UuD,DSD*
4-gram	*UUUU, UuDd, …, DuUs*
5-gram	*UUUUU, UuDds, …, DuUds*
Correlation	a_1a_2 , a_1a_3 , … , $a_{m-1}a_m$

(c) List of n-gram and correlation



(d) Correlation of multivariate attributes

Fig. 2. The preprocessing step in multivariate data classification

sub-patterns. Fig. 2(c) shows an example of n-gram based chunking. Moreover, in order to improve the classification accuracy, we exploit the inter-dependency structure that exists among the sensors (e.g., light and temperature), as illustrated in Fig 2(d).

We added the symbols representing the pairings of sensors that have a strong correlation (we used 0.6 as a threshold) into the list of original n-gram symbols as shown in the last row of Fig. 2(c). For example, if sensor a_1 and a_2 are correlated, we add a word, "a_1a_2", to the list. Once the data is transformed, we can simply treat them as a string of words and apply simple text classification algorithms to classify the data. In what follows, we will describe the details of the classification algorithms that we considered in our framework.

3.3 Supervised Methods

NBM (Naïve Bayes Model)**:** Bayesian classifiers are statistical classifiers and have exhibited high accuracy and speed when applied to a large database [11]. This technique chooses the highest posterior probability class using the prior probability computed from the training data set. For example, in the training phase, it learns the prior probability distribution such as, *P(uD|class=intrusion)* and *P(a_1a_2|class=normal)*, from the training data. In the test step, for each unlabeled window, a posterior probability is evaluated for each class C_i, as shown in (1). The test data is then assigned to class C_i for which $P(C_i|X)$ is the maximum.

$$P(C_i \mid X) = \frac{P(X \mid C_i)P(C_i)}{P(X)}, \text{ where } P(X \mid C_i) = \prod_{k=1}^{n} P(x_k \mid C_i) \text{ for } 1 \le i, j \le m,\ i \ne j \qquad (1)$$

SVM (Support Vector Machine)**:** This method is one of the most popular supervised classification methods. SVM is basically two-class classifier and can be extended for the multi-class classification (e.g., combining multiple one-versus-the-rest two-class classifiers). In our model, each window is mapped to a point in a high dimensional space, each dimension of which corresponds to an n-gram word or a correlation pair. For example, if a sliding window, W_i is {*uD*=2, *UUd*=10, *UDDD*=5, a_ia_{i+1}=0.8}, feature vector lists are {*uD*, *UUd*, *UDDD*, a_ia_{i+1}} and values according to the frequency factor are {0.2, 0.8, 0.6, 0.8}. The coordinates of the point are the frequencies of the n-gram words or coefficients of the correlation pairs in the corresponding dimensions. SVM learns, in the training step, the maximum-margin hyper-planes separating each class.

In testing step, it classifies a new window by mapping it to a point in the same high-dimensional space divided by the hyper-plane learned in the training step. For experiments, we used the Radial Basis Function (RBF) kernel [12], $K(x_i, y_i) = e^{-\gamma\|x_i - y_i\|^2}, (\gamma > 0)$. The soft margin allows errors during training. We set 0.1 for the two-norm soft margin value.

3.4 Unsupervised Methods

String-based Distance: This scheme measures the distance between two strings in order to measure the similarity. We can obtain the best matching class by comparing the feature vectors (standard vector space representations of documents) of each

known class with that of input data. Among many possible distance measures, we used two token-based string distance (Jaccard and TFIDF) and two edit-distance-based metrics (Jaro and Jaro-Winkler) that were reported to give a good performance for the general name matching problem in [14]. We briefly describe the metrics below. For details of each metric, refer to [13]. Using the terms of Table 1, the four metrics (2-5) can be defined as follows.

Table 1. Terms for string-based distance

Name	Descriptions	Name	Descriptions
x , y	n-grams and correlations for each sensor attribute.	$CC_{x,y}$	All characters in x common with y
C_x	All characters of x .	T_x	All n-gram and correlation terms for x .
$X_{x,y}$	# of transpositions of char in x relative to y		

$$\text{Jaccard}(x, y) = \frac{|T_x \cap T_y|}{|T_x \cup T_y|} \tag{2}$$

$$\text{TFIDF}(x, y) = \sum_{w \in T_x \cap T_y} V(w, T_x) \times V(w, T_y), \text{ where}$$

$$V(w, T_x) = log\left(TF_{w,T_y} + 1\right) \times \frac{log\left(IDF_w\right)}{\sqrt{\sum_{w'} \left(log\left(TF_{w,T_y} + 1\right) \times log\left(IDF_w\right)\right)}} \text{ (symmetrical for } V(w, T_y)\text{)}, \tag{3}$$

TF_{w,T_x} is the frequency of w in T_x , and IDF_w is the inverse of the fraction of names in a corpus containing w .

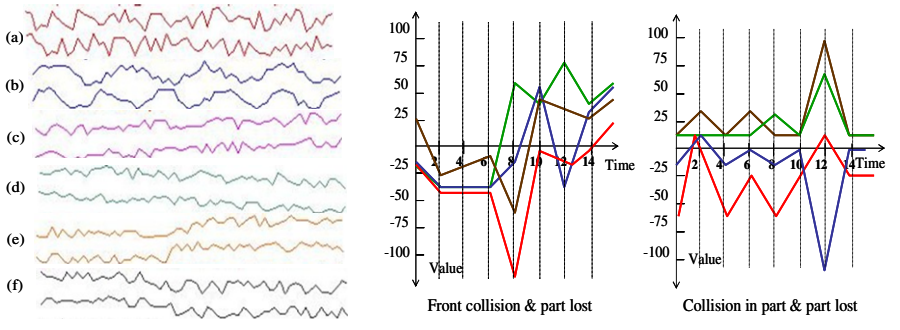
$$\text{Jaro}(x, y) = \frac{1}{3} \times \left(\frac{|CC_{x,y}|}{C_x} + \frac{|CC_{y,x}|}{C_y} + \frac{|CC_{x,y}| - X_{CC_{x,y}, CC_{y,x}}}{2|CC_{x,y}|} \right) \tag{4}$$

$$\text{Jaro-Winkler}(x, y) = Jaro(x, y) + \frac{max(|L|, 4)}{10} \times (1 - Jaro(x, y)), \text{ where } L \text{ is the longest common prefix of } x \text{ and } y \tag{5}$$

Vector-based Cosine Distance: This approach uses vector based distances to measure the similarity of the symbols. We model the n-gram symbols and correlation lists as vectors in the vector space. Each dimension of a vector corresponds to a unique term (i.e., an n-gram or an attribute pair for correlation) whose value consists of either a frequency of the term in the given window (if an n-gram) or the correlation coefficient of the two attributes (if an attribute pair). In order to measure the similarity of two vectors, we use a cosine distance, which is an angle between the two vectors, defined as: $Cos\theta = \frac{W_1 \bullet W_2}{\|W_1\| \bullet \|W_2\|}$ [11].

4 Experimental Results

In our experiment, we used two types of multivariate time series data obtained from [16]. Fig. 3(a) shows an example of the first data set containing six different classes of control patterns (Normal (a), Cyclic (b), Increasing trend (c), Decreasing trend (d), Upward shift (e), Downward shift (f)). Fig. 3(b) shows a fragment of the second data set which is robot traces containing force and torque measurements on a robot moving an object from one location to another. Each movement is characterized by 15 force/torque samples collected at regular time intervals starting immediately after failure detection. The trace data consists of 5 datasets, each of them defining a different learning problem labeled from LP1 to LP5 [16]. For experiments, we prepared a training data set that includes six different classes of control patterns and robot behavior classes such as normal, collision, and obstruction.



(a) Six different time series data (b) Examples of typical robot traces

Fig. 3. Data set: SCCTS and robot execution data

We performed k-fold cross-validation in order to evaluate the accuracy of each classification method. For the k-fold cross-validation, an input data set (S) is randomly partitioned into k mutually exclusive subsets ($S = \{S_1, S_2, \ldots, S_k\}$) of equal size. Training and testing is performed k times. In iteration i, the subset S_i is reserved as the test set, and the remaining subsets are collectively used to train the classifier. The accuracy of the classifier is then the overall number of correct classifications from the k iterations, divided by the total number of trials.

The result of experiments is shown in Fig 4. Fig 4(a) shows the accuracy of the six classifiers discussed in Section 3 using only the n-gram tokens and not considering the correlation tokens (see Fig 2(c).) Fig 4(b) shows the result using the both types of tokens. Different lengths of n-gram tokens are compared. For example, "3-gram" in the x-axis represents the classifications using only tokens up to length three (i.e., 1-3 grams).

As expected, the accuracy was gradually improved as longer tokens were taken into consideration. The longer tokens are likely to capture more temporal locality of patterns. The accuracy was generally higher when the correlation tokens were used along with the n-gram tokens. Noticeable improvements were observed in 3 and 4-gram experiments as shown in Fig. 4.

As expected, supervised methods (NBM and SVM) were more accurate than unsupervised methods. SVM showed the best performance among the tested methods. Among the unsupervised methods, classifiers using token-based string distance metrics were more accurate than the ones using edit-distance metrics. For this experiment, we used the classification library and package obtained from [17, 18].

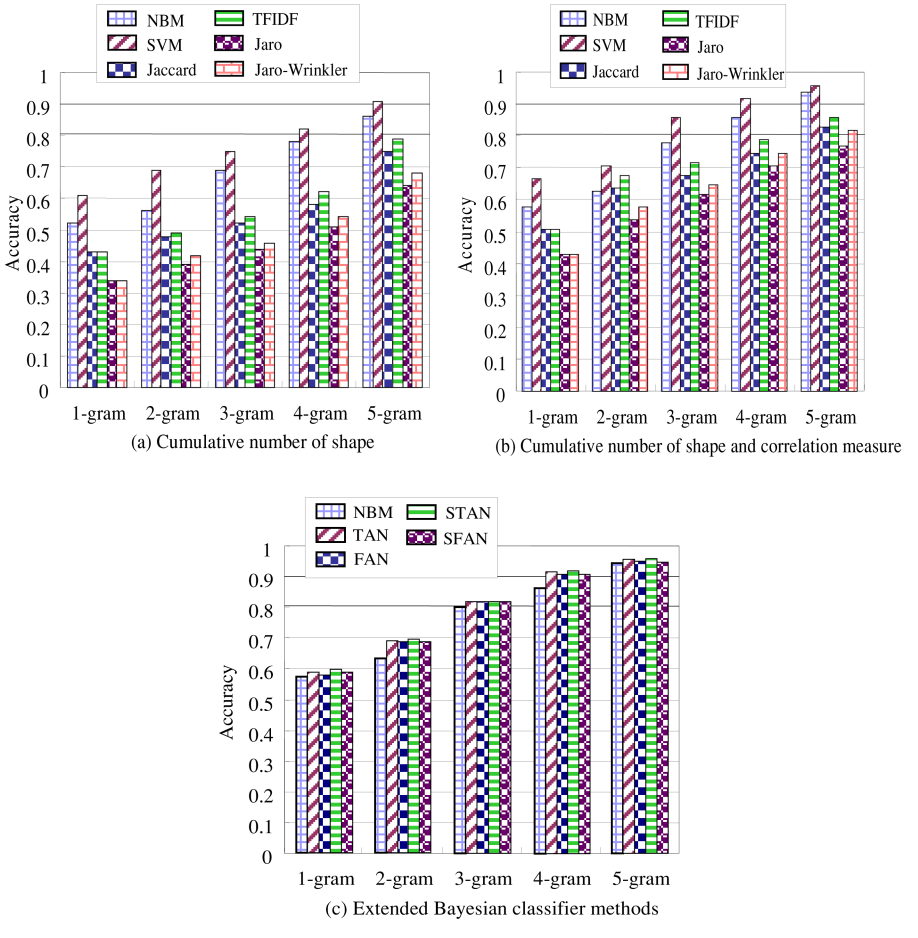


Fig. 4. Accuracy comparison (number of shapes and correlations between attributes)

Naïve Bayesian classifier in Fig. 4(a) assumes that the effect of an attribute on a given class is independent of the values of the other attributes. This assumption is called class conditional independence. However, attribute values of multivariate stream data collected from WSN may not be entirely independent from each others. For example, it is likely that the sensor readings of light and temperature would be correlated. In order to address this problem, in our experiment, we considered a set of extended Bayesian classifiers known to work well with correlated data, including TAN (Tree Augmented Naïve Bayes), FAN (Forest Augmented Naïve Bayes), STAN(Selective

Tree Augmented Naïve Bayes), and SFAN(Selective Tree Augmented Naïve Bayes) [15, 19]. Experimental results show that TAN and STAN method are better than the other methods as shown in Fig. 4(c). The result shows that the dependencies among attributes affect the classification accuracy for multivariate stream data.

5 Conclusion

In this paper, we proposed a scalable framework for multivariate stream data classification for continuous stream data. For classification, we employed the hierarchical piecewise linear representation to transform the continuous sensor streams into a discrete symbolic representation, which allows us to choose a classifier from a large pool of well-studied classification methods. We considered supervised methods including Naïve or extended Bayesian Model and SVM, and unsupervised methods including Jaccard, TFIDF, Jaro and Jaro-Winkler. In experimental results, SVM and TFIDF outperformed the other classification methods, and classification accuracy is higher when the correlations of attributes are also considered along with the n-gram token list.

Acknowledgement. This work was partially supported by the RRC Program of MOCIE and ITEP, and by ETRI (Telematics & USN Research Division) in Korea.

References

1. A. Mainwaring and J. Polastre, et al.: Wireless Sensor Networks for habitat monitoring. In WSNA (2002), pp.88-97
2. B. Xu and O. Wolfson.: Time-Series Prediction with Applications to Traffic and Moving Objects Databases. In MobiDE (2003), pp.56-60
3. R. C. Oliver and K. Smettem, et al.: Field Testing a Wireless Sensor Network for Reactive Environmental Monitoring. In ISSNIP (2004), pp.7-12
4. C. C. Aggrawal, J. Han, and P. S. Yu.: On Demand Classification of Data Streams. In KDD (2004), pp.503-508
5. M. W. Kadous and C. Sammut.: Classification of multivariate time series and structured data using constructive induction. Machine Learning Journal (2005), pp.176-216
6. H. Wang, W. Fan, P. S. Yu, and J. Han.: Mining Concept-Drifting Data Streams Using Ensemble Classifiers. In SIGKDD (2003), pp.226-235
7. J. Lin, E. Keogh, S. Lonardi, and B. Chiu.: A Symbolic Representation of Time Series with Implications for Streaming Algorithms. In DMKD (2003), pp.2-11
8. P. Geurts.: Pattern Extraction for Time Series Classification. PKDD (2001), pp.115-127
9. G. Xianping.: Pattern Matching in Financial Time Series Data. In Final Project Report for ICS 278 UC Irvine (1998)
10. R. Agrawal, G. Psaila, E. L. Wimmers, and Mohamed Zait.: Querying Shapes of Histories. In VLBD (1995), pp.502-514
11. J. Han and M. Kamber.: Data Mining Concepts and Techniques. Morgan Kaufmann Publishers (2000)
12. N. Cristianini and J. Shawe-Taylor.: An Introduction to Support Vector Machines. Cambridge University Press (2000)
13. W. W.Cohen, P. Ravikumar, and S. Fienberg.: A Comparison of String Distance Metrics for Naming-matching tasks. In IIWEB (2003)

14. B.W. On, D.W. Lee, J. W. Kang, and P. Mitra.: Comparative Study of Name Disambiguation Problem using a Scalable Blocking-based Framework. In JCDL (2005), pp.344-353
15. J. Chen and R. Greiner.: Comparing Bayesian Network Classifiers. In Proc. of UAI-99(1999), pp.101-108
16. S. Hettich and S. D. Bay.: The UCI KDD Archive [http://kdd.ics.uci.edu]. Irvine, CA: University of California, Department of Information and Computer Science
17. A Library for Support Vector Machines: http://www.csie.ntu.edu.tw/~cjlin/libsvm
18. SecondString (Jave-based Package of Approximate String-Matching): http://secondstring.sourceforge.net
19. Java Bayesian Network Classifier Toolkit: http://jbnc.sourceforge.net.