Multivariate Time Series Classification by Combining Trend-Based and Value-Based Approximations

Bilal Esmael[1], Arghad Arnaout[2], Rudolf K. Fruhwirth[2] and Gerhard Thonhauser[1]

[1] University of Leoben
8700 Leoben, Austria
Bilal@stud.unileoben.ac.at
Gerhard.Thonhauser@unileoben.ac.at

[2] TDE GmbH
8700 Leoben, Austria
{Arghad.Arnaout, Rudolf.Fruhwirth}@tde.at

Abstract. Multivariate time series data often have a very high dimensionality. Classifying such high dimensional data poses a challenge because a vast number of features can be extracted. Furthermore, the meaning of the normally intuitive term "similar to" needs to be precisely defined. Representing the time series data effectively is an essential task for decision-making activities such as prediction, clustering and classification. In this paper we propose a feature-based classification approach to classify real-world multivariate time series generated by drilling rig sensors in the oil and gas industry. Our approach encompasses two main phases: representation and classification.

For the representation phase, we propose a novel representation of time series which combines trend-based and value-based approximations (we abbreviate it as TVA). It produces a compact representation of the time series which consists of symbolic strings that represent the trends and the values of each variable in the series. The TVA representation improves both the accuracy and the running time of the classification process by extracting a set of informative features suitable for common classifiers.

For the classification phase, we propose a memory-based classifier which takes into account the antecedent results of the classification process. The inputs of the proposed classifier are the TVA features computed from the current segment, as well as the predicted class of the previous segment.

Our experimental results on real-world multivariate time series show that our approach enables highly accurate and fast classification of multivariate time series.

Keywords: Time Series Classification, Time Series Representation, Symbolic Aggregate Approximation, Event Detection.

1 Introduction

Multivariate time series data are ubiquitous and broadly available in many fields including finance, medicine, oil and gas industry and other business domains. The problem of time series classification has been the subject of active research for decades [1, 7].

The general time series can be defined as follow: A time series T is a series of ordered observations made sequentially through time. We denote the observations by:

$$x_i(t); [i = 1, \dots, n; t = 1, \dots, m] \text{ where:}$$

- i is the index of the different measurements made at each time point t,
- n is the number of variables being observed, and
- m is the number of observations made.

If the time series has only one variable ($n = 1$) then this time series is referred to as univariate, if it has two variables or more ($n > 1$) then it is referred to as multivariate. One example of multivariate time series is drilling rig data; where many mechanical parameters such as torque, hook load and block position, are continuously measured by rig sensors and stored in real time in the databases. Fig. 1 shows drilling multivariate time series consisting of eight variables.

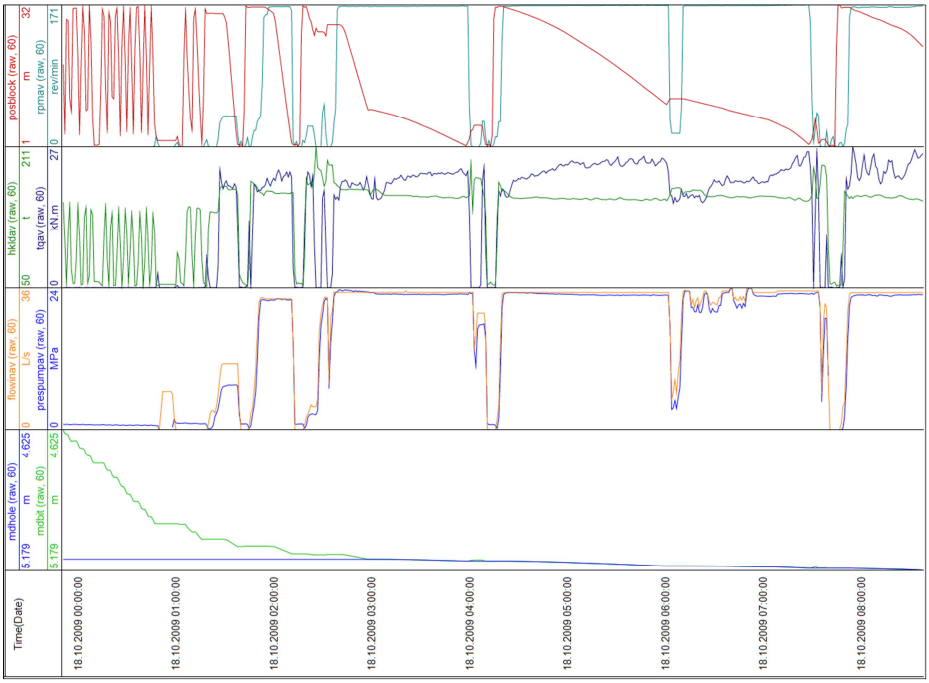

Fig. 1. A multivariate time series of drilling data. This time series consists of eight variables representing eight mechanical parameters measured at the rig.

Multivariate time series classification is a supervised learning problem aimed for labeling multivariate series of variable length. Time series classification can be divided into two types. In the first type (simple classification) each time series is classified into only one class label, whereas in the second type (strong classification) each time series is classified into a sequence of classes.

This work focuses on the second type of classification. Our approach aims to classify multivariate time series (like the one shown in Fig. 1) into a sequence of operations or classes $\mathrm{op}_1(\mathrm{st}_1, \mathrm{et}_1), \ldots, \mathrm{op}_n(\mathrm{st}_n, \mathrm{et}_n)$ where st_i and et_i represent the start time and end time of the operations respectively. Fig. 2 shows the result of such a classification process.

Fig. 2. A sequence of 10 operations with different durations.

The main contributions of this work are:

- An approach to represent time series by combining value-based and trend-based approximations (TVA). It extends Symbolic Aggregate Approximation (SAX) [2] by adding new string symbols (U, D and S) to represent the directions of the time series.
- A memory-based classifier for multivariate time series classification. The classifier is trained with the TVA features extracted from our representation. In addition, it uses the previous predicated class as an additional feature to predicate the class of the current segment.

The remainder of the paper is organized as follows: Section 2 introduces the state-of-art techniques for time series representation. Section 3 presents the general framework of our approach. Section 4 explains the details of TVA representation. Section 5 discusses the time series classification. Finally, section 6 presents the experimental results of the proposed approach using real-world data from the drilling industry, and Section 7 concludes the work.

2 State of the Art

Time series datasets are typically very large. The high dimensionality, high feature correlation, and the large amount of noise that can be present in time series, pose a challenge to time series data mining tasks [2]. The high dimensionality of such time series increases both the access time to the data and computation time needed by the

data mining algorithms used [8]. Additionally, visualization techniques need to employ data reduction and aggregation techniques to cope with the high volume of data that cannot be plotted in details at once. Furthermore, the very meanings of terms such as “similar to” and “cluster forming” become unclear in high dimensional space [1].
The aforementioned reasons make applying machine learning techniques directly on raw time series data cumbersome. To overcome this problem, the original “raw” data need to be replaced by a higher-level representation that allows efficient computation on the data, and extracts higher order features [2, 3 and 4].
Several representation techniques, known as dimensionality reduction techniques, have been proposed. This includes the Discrete Fourier Transform (DFT), the Discrete Wavelet Transform (DWT), Piecewise Linear Approximation (PLA), Piecewise Aggregate Approximation (PAA), Adaptive Piecewise Constant Approximation (APCA), Singular Value Decomposition (SVD) and Symbolic Aggregate Approximation (SAX). Choosing the appropriate representation depends on the data at hand and on the problem to be solved. Furthermore, it affects the ease and efficiency of time series data mining [1].

Trend-based and value-based approximations have been used extensively in the last decade. Kontaki et al. [10] propose using PLA to transform the time series to a vector of symbols (U and D) denoting the trend of the series. Keogh and Pazzani [8] suggest a representation that consists of piecewise linear segments to represent a shape; and a weight vector that contains the relative importance of each individual linear segment.
SAX, proposed by Lin et al. [2], is a symbolic approximation of time series. It employs a discretization technique that transforms the numerical values of the time series into a sequence of symbols from a discrete alphabet. The discretization process allows researchers to apply algorithms from text processing and bioinformatics disciplines [2]. SAX has become an important tool in the time series data mining, and has been used for several applications such as time series classification, events detection [5, 6], and anomaly detection [11]. It enables using the Euclidian distance of the discretized subsequences [9], and allows both dimensionality reduction and lower bounding of L_p norms [11].
Although the above mentioned advantages, SAX suffers from some limitations. It does not pay enough attention to the directions of the time subsequences and may produce similar strings for completely different time series. To overcome this problem we propose the TVA representation which extends SAX by adding new string symbols in order to represent the trends of time series.

3 Our Approach

The general framework of the proposed approach is shown in Fig. 3. The given multivariate time series is first divided into a sequence of smaller segments by sliding a window incrementally across the time series. Then, the processing is performed in two phases: representation and classification

- In the representation phase each segment w_i is represented by a pair of characters (v, t). The first character v represents the linguistic value of the time series and takes one of these values: (a = low), (b = normal), (c = high), etc. The second character t describes the local trend of the time series and takes one of these values: (U = up), (D = down) or (S = straight).
- In the classification phase, a memory-based classifier is trained and used to assign a class label to each segment.

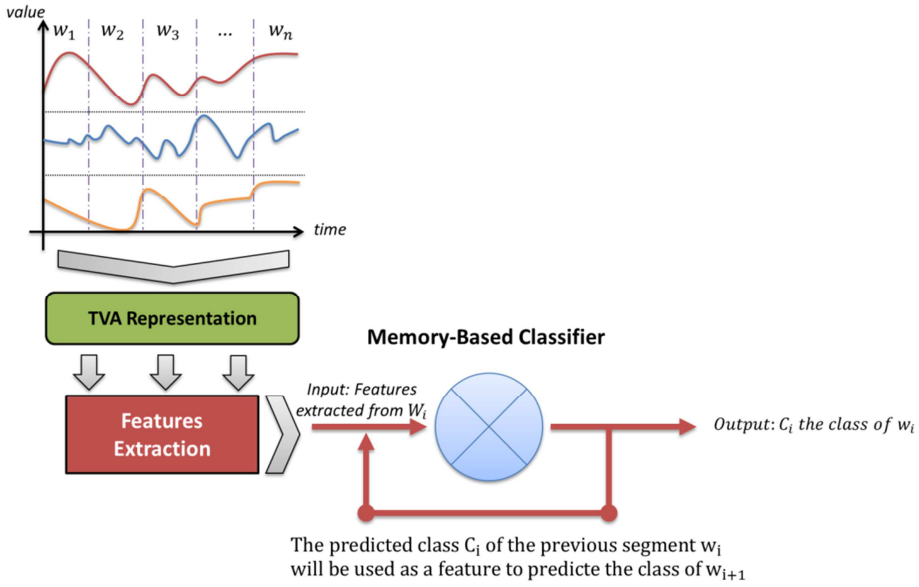

Fig. 3. The general framework of the proposed approach

4 TVA Representation

In the classification phase, we are not interested in the exact numerical values of each data point in the given time series. What we are interested in are the trends, shapes and patterns existing in the data. To recognize these patterns first it is required to discover the simple local trends such as “increase in the hookload” and “decrease in the torque” and to divide the numerical values of the time series into discrete levels such as “high hookload” and “low pressure”.

The TVA representation transforms the numerical values of each variable in the given time series into a sequence of $< value, trend >$ pairs. The multivariate time series T is hence transformed as follows:

$$\text{T}=\begin{bmatrix} x_1^1 & x_1^2 & \cdots & x_1^m \\ x_2^1 & x_2^2 & \cdots & x_2^m \\ \vdots & \vdots & \cdots & \vdots \\ x_n^1 & x_n^2 & \cdots & x_n^m \end{bmatrix} \Rightarrow R = \begin{bmatrix} (v_1^1, t_1^1) & (v_1^2, t_1^2) & \cdots & (v_1^s, t_1^s) \\ (v_2^1, t_2^1) & (v_2^2, t_2^2) & \cdots & (v_2^s, t_2^s) \\ \vdots & \vdots & \cdots & \vdots \\ (v_n^1, t_n^1) & (v_n^2, t_n^2) & \cdots & (v_n^s, t_n^s) \end{bmatrix}$$

where R is the matrix that contains the $< value, trend >$ pairs, s denotes the number of the segments, v_i^k represents the discrete level of the time series variable i in segment k, and t_i^k represents the trend (direction) of this variable in the segment.

4.1 Value-Based Approximation:

In our TVA representation we use the SAX technique to approximate the values of the time series. Two steps should be followed:

- Transforming the given time series T into PAA segments.
- Discretization of the time series based on predefined breakpoints.

Transforming Step

In this step, PAA is used to transform the given time series T of length m into a time series of length s by dividing the original time series into equal-sized segments, and then computing the mean value w_i for each segment i as follows:

$$w_i = \frac{s}{m} \sum_{j=\frac{m}{s}(i-1)+1}^{\frac{m}{s}i} x_j$$

The time series T is represented by a vector of mean values $W = \{w_1, \dots, w_s\}$

Discretization Step

In this step, a further transformation is applied to obtain a discrete representation by producing symbols with equiprobability. The inventors of SAX mentioned that in empirical tests on more than 50 datasets, the normalized subsequences have a highly Gaussian distribution [2]. This enables determining the "breakpoints" that produce equal-sized areas under a Gaussian probability density function. After determining the breakpoints, the time series T is discretized in the following manner: All PAA coefficients that are below the smallest breakpoint are mapped to the symbol "a", all coefficients greater than or equal to the smallest breakpoint and less than the second smallest breakpoint are mapped to the symbol "b", and so forth.

Fig. 4 illustrates how the transformation and discretization phases are applied on the data (hook load data). In this example, with m = 100 and s = 10, the given time series is mapped to the word *hcdacafgfg*

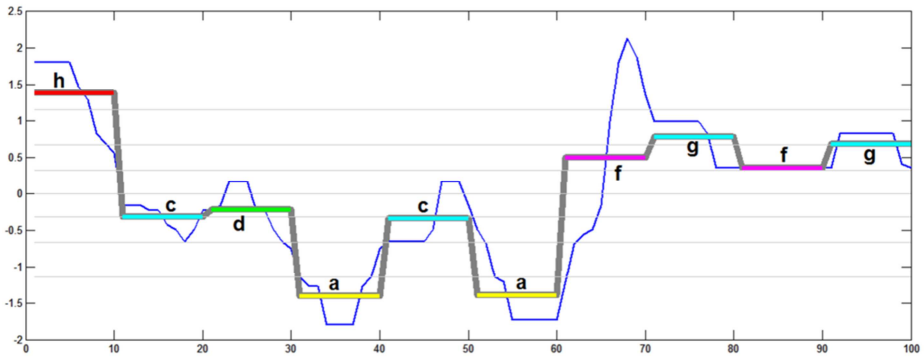

Fig. 4. A time series (blue line) is discretized by first obtaining a PAA approximation (gray line) and then using predetermined breakpoints to map the PAA coefficients into symbols.

Indeed, representing the time series, using only the value approximation (SAX), causes a high possibility to miss some important patterns in some time series data. SAX does not pay enough attention to the shapes of the time subsequences and may produce similar strings for completely different time series. Fig. 5 shows an example.

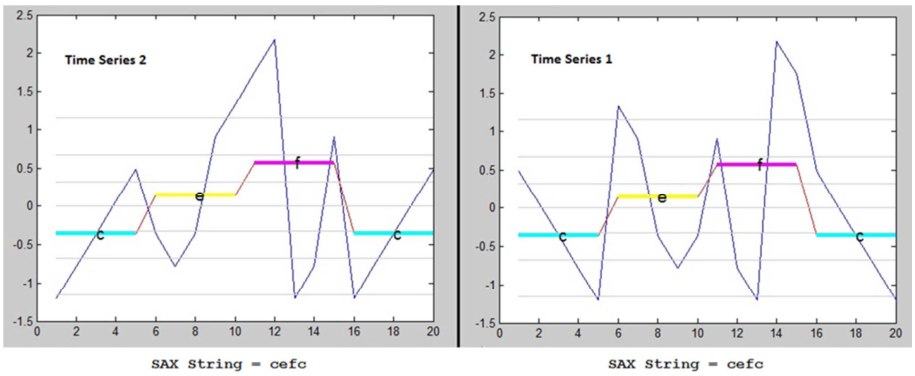

Fig. 5. Two completely different time series that have the same sax string

The above mentioned problem is overcome by adding trend-based approximation beside value-based approximation in order to represent the directions of time series.

4.2 Trend-Based Approximation

We propose using the trends as basis for classifying time series data because these trends form an important characteristic of a time series. In addition, trend-based approximation of time series is closer to human intuition [10].

To generate a trend-based approximation, the least squares method is used to fit a straight line through the set of data points. The least squares method assumes that the best-fit line is the line that has the minimal sum of the squared deviations (least squares error) from a given set of data.
According to the least squares method, the best fitting line has the property that:

$$\sum_{i=1}^{n} d_i^2 = \sum_{i=1}^{n} [y_i - f(x_i)]^2 \rightarrow a\ minimum$$

After constructing the lines that fit the data points, the slopes of these lines is calculated, and finally the trend characters U, D or S are computed based on the value of the slope. Fig. 6 illustrates how the above mentioned steps are applied to construct the trend approximation for a part of hook load data.

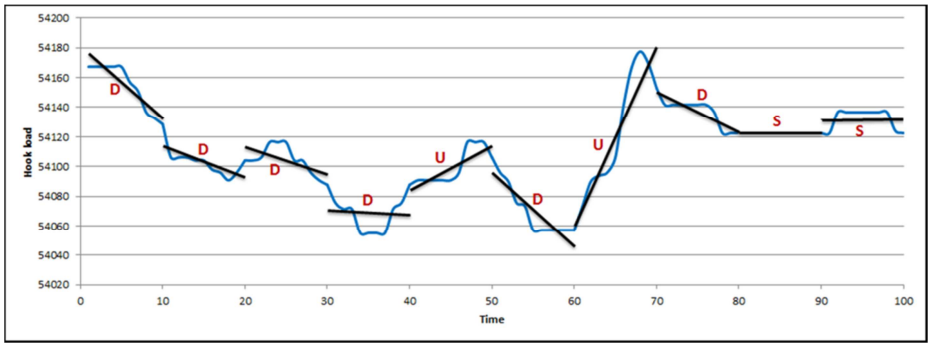

Fig. 6. Trend-based approximation

Using both, the trend-based and the value-based approximation (TVA representation), the given time series is mapped to the string "*hDcDdDaDcUaDfUgDfSgS*". Lower case is used to represent SAX values, and upper case is used to represent trends.

5 Time Series Classification

The classification phase starts by extracting a set of features from the TVA representation of the given time series. A feature-vector of fixed length is created for each segment of the time series. The vector has the following form:

$$F^k = (< v_1^k, t_1^k >, < v_2^k, t_2^k >, \dots, < v_n^k, t_n^k >),$$

Where v_i^k and t_i^k represents the SAX character and trend character respectively, and n represents the number of the variables in the given time series. For example, if the given time series has 10 variables and is divided into s segments, then the extracted dataset will have 20 features (columns) and s instances (rows) as shown in Table 1.
After extracting the features, the classifier is trained to assign a class (label) to each segment. In this work we propose a memory-based classifier which takes the previous

output of the classification process into consideration. The inputs of the proposed classifier are the feature-vector F^k of the current segment k as well as the predicted class $C^{k-1}$of the previous segment. Fig. 7 illustrates the pseudo-code of the classification algorithm.

Table 1. An example of the extracted dataset

Seg#	$value_1$	$trend_1$	$value_2$	$trend_2$	…	$value_{10}$	$trend_{10}$
1	v_1^1	t_1^1	v_2^1	t_2^1	…	v_{10}^1	t_{10}^1
2	v_1^2	t_1^2	v_2^2	t_2^2	…	v_{10}^2	t_{10}^2
…	…	…	…	…	…	…	…
s	v_1^s	t_1^s	v_2^s	t_2^s	…	v_{10}^s	t_{10}^s

Classification Algorithm

Input:

- A multivaite time series T of length m

Output:

- A sequence of labels (classes)

Do

- Create an empty sequence of classes SC.
- Divide the original time series T into a set W of smaller equal-sized segments, where $W = \{w_1, w_2, \dots, w_s\}$
- For each segment in W
    - Represent the current segment w_i as mentioned in section 4.
    - Create the feature-vector F_i
    - Get the predicted class of the previous segment c_{i-1}
    - Call the prediction method **predict (F_i, c_{i-1})** which returns the class c_i of the current segment.
    - Add the predicated class c_i to the sequence SC.
- End For
- For all classes in SC
    - Combine the consecutive equal classes $c_1, \dots, c_r$ in one class C.
    - Set the start time of C equal to the start time of the first class c_1
    - Set the end time of C equal to the end time of the last class c_r
- End For All
- Return SC

End

Fig. 7. The classification algorithm

Although the memory-based classifiers are simple, as we will show, they improve the classification accuracy significantly. The experimental results show that the average improvement in accuracy is about 8% compared to a traditional classifier.

Many classification techniques can be used to classify the time series. In this work we tested Naïve Bayes, Support Vector Machine, Rule Induction, K-Nearest Neighbor and Decision Trees. Using all these techniques, the classification accuracy was high as we show in the next section. Also, the training time is significantly reduced because the number of extracted features is small.

6 Experimental Results

To evaluate our approach, we tested it with real-world data. Two time series were used in our experiments. Table 2 illustrates these two time series:

Table 2. Time series parameters

Time Series	Length	Frequency	#Variables	#Classes
1	376,840	0.1 Hz	12	10
2	195,808	0.2 Hz	10	9

For all experiments a sequence of ten classes to the first time series, and a sequence of nine classes to the second time series was assigned. Each one of these classes represents one particular operation during drilling. The final output of the classification task is similar to Fig. 2. The proposed approach to represent the data was applied to create the feature space in a first step. Following that, RapidMiner [12] and LIBSVM [13] were used to test the classifiers using the cross validation technique.

Table 3: Classification accuracy

	Time Series #1		Time Series #2	
Window Size	Traditional classifier [%]	Memory-based classifier [%]	Traditional classifier [%]	Memory-based classifier [%]
2	80.40	**90.87**	92.38	**97.88**
3	78.23	89.58	93.10	97.71
4	76.62	87.23	92.84	97.51
5	72.87	83.55	93.82	97.67
6	71.40	82.61	93.54	97.43
7	71.03	81.85	93.50	97.14
8	70.25	82.01	92.92	96.70
9	69.86	81.10	92.64	96.35
10	69.41	81.28	92.45	96.33

To measure the improvement that the memory-based classifier provides, two types of classifiers were trained and tested with a varying window size. The first classifier was trained only with TVA features. The second classifier (memory-based classifier) was

trained using the extracted features as well as the previously predicted classes as input. Table 3 and Fig. 8 show the results of the Naïve Bayes classifier.

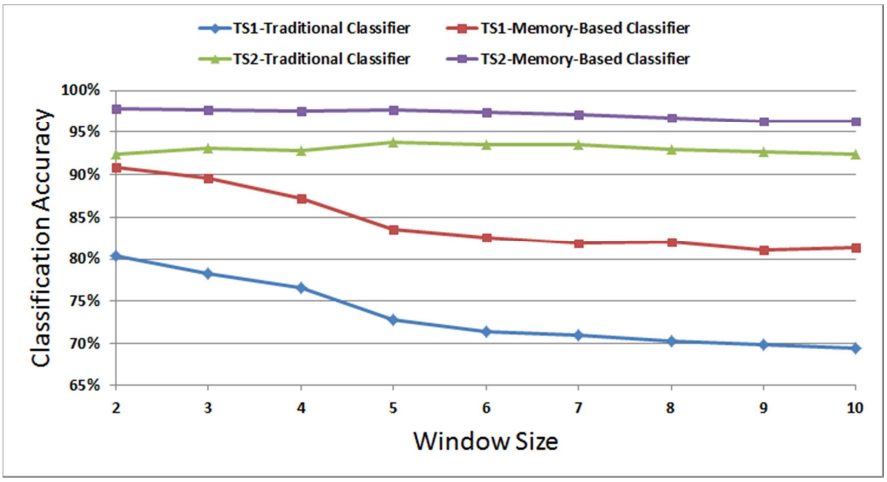

Fig. 8. Classification accuracy vs. window size

Table 4 shows the confusion matrix of one of the experiments in which the Naïve Bayes classifier is applied to time series #2 using a window size of 2.

Table 4. Confusion matrix

	MoveUP	MoveDN	MakeCN	CircHL	ReamDN	DrlRot	ReamUP	WashDN	WashUP	Precision [%]
MoveUP	343	8	30	0	0	0	17	0	2	**85.7**
MoveDN	16	442	4	0	0	0	1	1	1	**95.0**
MakeCN	12	7	1949	8	0	0	0	20	7	**97.3**
CircHL	11	1	0	4636	69	27	22	25	61	**95.5**
ReamDN	1	0	1	13	2155	7	24	30	12	**96.0**
DrlRot	0	0	0	2	5	19028	1	3	1	**99.9**
ReamUP	5	0	0	5	32	54	2298	1	22	**95.0**
WashDN	0	6	8	7	19	2	1	920	46	**91.1**
WashUP	14	1	5	5	2	1	11	27	1632	**96.1**
Recall [%]	**85.3**	**95.0**	**97.6**	**99.1**	**94.4**	**99.5**	**96.7**	**89.5**	**91.4**	

In addition to Naïve Bayes, four classification techniques were tested. These techniques are: Support Vector Machine (SVM), Rule Induction (RI), Decision Trees (DT) and K-Nearest Neighbor (K-NN). Table 5 summarizes the results.

Table 5. The classification accuracies of different techniques

	SVM	RI	DT	K-NN
Time Series#1	92.9%	91.12%	90.0%	92.8%
Time Series#2	95.23%	98.24%	94.11%	96.9%

7 Conclusion and Future Work

The following conclusion can be drawn from the concepts presented in this paper:

- Representing multivariate time series by combining both the value-based and trend-based approximations leads to reduce the dimensionality of the time series largely.
- The reduced representation can be used as alternative to the time series without losing any important characteristics or patterns exist in the original time series data.
- Memory-based classifiers can improve the classification accuracy of the time series significantly.

Our aim for future work is to improve this approach and use it for writing reports automatically. TVA will be used as an intermediate representation between the numerical values of time series and the human language.

Acknowledgment

We thank TDE Thonhauser Data Engineering GmbH for supporting this work and for the permission to publish this paper.

References

1. Ratanamahatana, C.A., Lin J., Gunopulos D., Keogh E., Vlachos M., and Das G.: Data Mining and Knowledge Discovery Handbook 2010. 2nd Edition. Eds. O. Maimon, L. Rokach. Springer. Pages 1049-1077, (2010)
2. Lin, J., Keogh, E., Lonardi, S. & Chiu, B.: A Symbolic Representation of Time Series, with Implications for Streaming Algorithms. In proceedings of the 8th ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery. San Diego, CA. June 13 (2003)
3. Batal, I., Valizadegan H., Cooper GF., Hauskrecht M.: A Pattern Mining Approach for Classifying Multivariate Temporal Data, IEEE International Conference on Bioinformatics and Biomedicine , Atlanta, Georgia, November (2011)
4. Batal I., Sacchi L., Bellazzi R., Hauskrecht M.: Multivariate Time Series Classification with Temporal Abstractions. In Proceedings of the Twenty-Second International Florida AI Research Society Conference (FLAIRS 2009), May (2009)
5. Onishi, A., Watanabe C.: Event Detection using Archived Smart House Sensor Data obtained using Symbolic Aggregate Approximation. PDPTA (2011)

6. Zoumboulakis M., Roussos G.: Complex Event Detection in Wireless Sensor Networks. In Proceedings of 2nd European Conference on Smart Sensing and Context (EuroSSC), Lake District, UK (2007)
7. Wei, L., Keogh, E.: Semi-Supervised Time Series Classification. The Twelfth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, SIGKDD (2006)
8. Keogh, E., Pazzani, M.: An enhanced representation of time series which allows fast and accurate classification clustering and relevance feedback. In 4th International Conference on Knowledge Discovery and Data Mining. New York, NY, Aug 27-31. pp 239-243 (1998)
9. Hung, N.Q.V., Anh, D.T.: Combining SAX and Piecewise Linear Approximation to Improve Similarity Search on Financial Time Series. Proceedings of the 2007 IEEE International Symposium on Information Technology Convergence (ISITC 2007), Jeonju, Korea, (2007)
10. Kontaki, M., Papadopoulos, A.N., Manolopoulos, Y.: Continuous Trend-Based Classification of Streaming Time Series, Proceedings of the 9th East-European Conference on Advances in Databases & Information Systems (ADBIS'2005), pp. 294-308, Tallinn, (2005)
11. Keogh, E., Lin, J., Fu A.: HOT SAX: Efficiently Finding the Most Unusual Time Series Subsequence. In Proceeding of the 5th IEEE International Conference on Data Mining (ICDM 2005), pp. 226 - 233., Houston, Texas, Nov 27-30, (2005)
12. Mierswa, I., Wurst, M., Klinkenberg, R., Scholz, M., Euler, T.: YALE: Rapid Prototyping for Complex Data Mining Tasks, in Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-06), (2006)
13. Chih-Chung, C., Chih-Jen, L.: LIBSVM : a library for support vector machines. ACM Transactions on Intelligent Systems and Technology, 2:27:1--27:27, (2011). Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm