Available online at www.sciencedirect.com

ScienceDirect

CAAI Transactions on Intelligence Technology 1 (2016) 61–71
http://www.journals.elsevier.com/caai-transactions-on-intelligence-technology/

Original article

Exploiting structural similarity of log files in fault diagnosis for Web service composition

Xu Han [a,*], Binyang Li [b], Kam-Fai Wong [c], Zhongzhi Shi [d]

[a] *Beijing Key Laboratory of Electronic System Reliability Technology, Capital Normal University, Beijing, China*
[b] *University of International Relations, Beijing, China*
[c] *Department of Systems Engineering & Engineering Management, The Chinese University of Hong Kong, Hong Kong, China*
[d] *The Key Laboratory of Intelligent Information Processing, Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China*

Available online 7 June 2016

Abstract

With increasing deployment of Web services, the research on the dependability and availability of Web service composition becomes more and more active. Since unexpected faults of Web service composition may occur in different levels at runtime, log analysis as a typical data-driven approach for fault diagnosis is more applicable and scalable in various architectures. Considering the trend that more and more service logs are represented using XML or JSON format which has good flexibility and interoperability, fault classification problem of semi-structured logs is considered as a challenging issue in this area. However, most existing approaches focus on the log content analysis but ignore the structural information and lead to poor performance. To improve the accuracy of fault classification, we exploit structural similarity of log files and propose a similarity based Bayesian learning approach for semi-structured logs in this paper. Our solution estimates degrees of similarity among structural elements from heterogeneous log data, constructs combined Bayesian network (CBN), uses similarity based learning algorithm to compute probabilities in CBN, and classifies test log data into most probable fault categories based on the generated CBN. Experimental results show that our approach outperforms other learning approaches on structural log datasets.

Keywords: Web services composition; Fault diagnosis; Combined Bayesian network (CBN); Similarity; Probability

1. Introduction

In recent years, as a promising computing paradigm, service-oriented computing (SOC) [1] has changed the way of design, delivery and consumption of software applications. Web services technology aiming at implementing service oriented architecture has been widely applied to different areas for research or business purposes. Accordingly, more and more business functions are published as Web services (WS) by various organizations and companies. There are two main approaches of developing Web services, including SOAP-based Web services [2] and RESTful Web services [3,4]. Numerous atomic Web services can be regarded as access points for applications without relying on other Web services. When a user request cannot be fulfilled by atomic Web services, composite Web service plays an important role in providing complex collaboration and interaction between multiple Web services.

During the past ten years, a large number of existing standards [5] for Web service composition have been defined. Web service orchestration language is one of these standards for describing executable business processes, which are composed of Web services. Based on OASIS standards, Web Services Business Process Execution Language (WSBPEL or BPEL) [6] is an executable orchestration language for modeling business processes with Web services. Depending on Web service composition, business processes can be executed by the support of BPEL engine from a third party. Some recent

* Corresponding author.
E-mail addresses: hanxu@cnu.edu.cn (X. Han), byli@uir.cn (B. Li), kfwong@se.cuhk.edu.hk (K.-F. Wong), shizz@ics.ict.ac.cn (Z. Shi).
Peer review under responsibility of Chongqing University of Technology.

http://dx.doi.org/10.1016/j.trit.2016.03.006

research work has focused on extending BPEL engine to enable composition of SOAP-based, RESTful and OSGi services [7]. Since there are more and more Web services (esp. RESTful Web services) used in practice, the complexity of Web service composition is becoming higher than before. For this reason, how to ensure dependability and availability is essential for Web service composition. To enhance the dependability of service flow execution, fault tolerant Web service orchestration [8] was proposed. As a critical aspect in fault tolerance framework, fault diagnosis aims at identifying or locating the causes with high probability to explain process exceptions and failures based on runtime information. Since most detailed running information in processes execution is recorded in log files, fault diagnosis by learning from log data is becoming an important issue in this area.

Currently, semi-structured data formats, including XML and JSON, are used as the standards of information representation on the Internet. Due to its good flexibility and interoperability, more and more log files of software running information are represented using the XML/JSON format, especially for Web services. Thus, it becomes a key topic of fault diagnosis research which focuses on analyzing semi-structured and XML/JSON like log. Generally, semi-structured documents have much richer structural information than flat ones, which has potential influence on classification accuracy. Taking this into account, the main task of learning from this kind of documents will have more challenges than before. However, for most classification methods of log analysis, IR-based methods are commonly used ignoring a significant amount of structural information, which leads to low classification accuracy. Therefore, how to learn the structural information from the log has great impact on the accuracy of fault classification.

In this paper, we propose a similarity-based Bayesian learning approach for fault classification of semi-structured logs. Our method is to first estimate similarity degrees of structural elements from heterogeneous log data. Then the basic structure of combined Bayesian network (CBN) is constructed, and the similarity-based learning algorithm is used to compute probabilities in CBN. Finally, test log data can be classified into most probable fault categories based on the generated CBN.

The rest of this paper is organized as follows. Section 2 introduces the related work concerning fault diagnosis of Web service composition and existing classification methods for (semi-)structured documents. Section 3 provides an overview of the similarity based structural classification approach and the CBN model. The details of CBN generation, including how to compute probabilities of CBN using similarity-based learning algorithm, are presented in Section 4. Experimental results of this approach compared to other learning approaches are shown in Section 5. Finally, Section 6 draws the conclusion.

2. Related work

Over the past years, some research work in Web services area has concentrated on how to enhance dependability and availability of Web services. Fault tolerant Web services orchestration [8] is supported by fine grained identification of exception and fault causes and the consequent execution of effective exception and fault handlers [9,10]. As an important step of fault tolerance, fault diagnosis has attracted wide attention of academic community increasingly. From the methodology perspective, the existing work on fault diagnosis in this area can be divided into two main categories, including model-based diagnosis and data-driven diagnosis.

With respect to model-based approaches, the basic idea is to model the behavior and inner logic of the diagnosed service, and then discover runtime faults based on its model. The on-going work has been described in some published papers. WS-DIAMOND [11] is a European research project which eight research agencies have participated in. In this project, model-based diagnosis is adopted as the principal approach. Yan et al. [12,13] presented a model-based approach for diagnosing orchestrated Web service processes. In their approach, Web services with faults can be deduced from the variable dependency on execution trajectory, which is represented by the generated automata of BPEL description. With the assumption that behavioral descriptions of individual activities may not be totally given, Mayer et al. [14] presented an approach of isolating minimal sets of faulty activity executions based on the process structure. Considering composite service adaption to the dynamic execution environment, Dai et al. [15] analyzed the error propagation relation between any two services and gave uncertain casual relation between exceptions and services by computing error propagation degree. In addition, some research groups proposed testing frameworks [16] and hybrid models [17] for fault diagnosis of Web service composition.

For data-driven approaches, the diagnosis problem is usually transformed into the classification problem. Then it can be solved by using data mining and machine learning algorithms on log file data. As we know, many reported research efforts have focused on mining log files of computing systems. And there are some common places in log mining methods for regular computing systems and Web services. For this reason, we can make reference to those existing methods in the related area. Considering the differences between two basic data types — plain text data and semistructured data, mining approaches are often designed and implemented according to the type of training and test data. For plain-text log data, Li et al. used Bayes method [18,19] to categorize text messages in log files, and utilize the temporal information to improve classification performance. In Ref. [20], Bayes classifier, semi-supervised learning, and decision trees, are used to automatically recognize symptoms of recurrent faults. As for semi-structured log data, there are also some corresponding classification approaches. In Ref. [21], a database of failure signatures against which undiagnosed failure data can be matched, is constructed from monitoring data. And then anomaly-based clustering method is proposed to generate right clusters for diag- nosing failures with low-confidence match. Denoyer and Gallinari [22] provided a generative model for classification task based on Bayesian networks, which can handle both structure and content. Zaki and Aggarwal [23] presented an effective rule based classifier for XML data using frequent discriminatory

substructures within XML documents. As a Bayesian learning approach of semi-structured data, our approach is similar to that given by Denoyer and Gallinari in methodology aspect. But they are different in nature, because their generative model only concerns the training data in identical logic structure, whereas our approach aims at constructing the CBN model based on log data in different structures from heterogeneous sources (which will be discussed later in detail).

In addition, there are other diagnosis mechanisms and frameworks for Web services. In Ref. [24], a self-healing plug-in for BPEL engine is presented, which can enhance the ability of a standard engine to provide process-based recovery actions. It only provides the self-healing mechanism at infrastructure level without referring to diagnosis methods in detail. Ardissono et al. [8] proposed a framework for Web Service orchestration which employs diagnostic services to support a fine grained identification of exception causes.

3. A Bayesian network approach to fault diagnosis

As a data-driven approach, the task of fault diagnosis is to construct the generative model based on the training data from semi-structured log files, and then classify test log data into possible fault categories using this model. On account of the heterogeneity of log file sources in practice, it is a new challenge to develop methods of learning from the semi-structured log data, which are similar in content but not identical in structure. For this reason, it is important to exploit the structural information contained in XML/JSON documents for learning task. For sake of simplicity, we take XML log files as example in this paper. According to the classical representation, an XML document can be considered as a tree in which each node represents a structural element. Given two XML log files generated by different service execution engines, the corresponding schema trees can be extracted from them, which often have similar content but slightly different structures. In this paper, we extend our previous work [25] by proposing an efficient online learning algorithm for the dynamic environment. Our main idea is to find the similarity between the nodes of these two trees, and then construct the generative model by learning from the training log data based on similarity degrees. Herein, we propose combined Bayesian network as a generative model for semi-structured log data, the probabilities of which depend on training data distribution of both corresponding elements and their counterparts with high similarity degrees.

3.1. Overview of the approach

In this approach, the learning problem is simplified by limiting the number of XML log files to two, which constitute the training dataset. (If the number of log files is more than two, the learning problem can be transformed into the equivalent one which consists of several two-file problems.) For example, suppose there are two log files F_0 and F_1 as shown in Fig. 1. We can find that F_0 and F_1 have similar content but slightly different structures. In view of their structural differences, generating Bayesian network model from the training dataset is non-trivial. Herein, how to handle the similarity relation between schema elements of F_0 and F_1 can be taken as the key point. The objective of our approach is to generate the CBN by learning from all relevant data in training dataset based on quantified similarity.

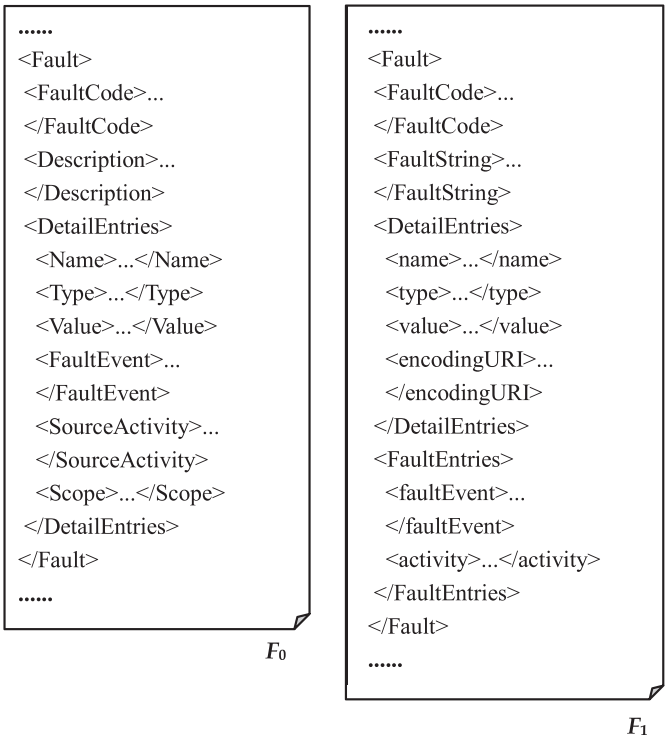


Fig. 1. Fragments of two sample XML log files.

To give an overview of the fault diagnosis approach, we will firstly introduce its principal steps. Suppose the schema trees T0 and T1 are extracted from XML log files F0 and F1, then the main steps in this approach can expressed as follows:

1. Estimate the normalized similarity degrees of the nodes of T_0 and T_1, and then find the similar node pairs by ignoring those pairs whose degrees are under the threshold value.
2. Create the basic structure of combined Bayesian network, which includes combination part and private part.
3. For each schema tree, calculate the combination similarity degrees between its nodes and corresponding ones in CBN, and normalize the values of combination similarity degrees.
4. Compute the probabilities of the constructed CBN on all relevant data in training dataset using the similarity degrees obtained from steps 1 and 3.
5. Given a runtime log record in XML format, use the generative CBN model to classify the test log data, and diagnose possible faults according to the category it falls in.

As we see, steps 1, 3 and 4 play important role in this approach, which are responsible for the computation of similarity degrees. The detailed discussion of these steps will be given in Section 4.

3.2. Modeling semi-structured log files with combined Bayesian networks

The Bayesian network is a suitable model for representing the dependencies and relations between different elements of semi-structured data. On account of structural differences of training data, we propose combined Bayesian network, which is modeled by learning from heterogeneous training data based on quantified similarity. As a generative model, CBN is capable of handling both structure and content information, and can be used to classify test log data.

Generally, each log file consists of a set of log data records, which will be labeled with related fault categories. These labeled log records constitute the training dataset. We associate a CBN model to each category of the training dataset. Since data records in same category may have different logical structures, a CBN is constructed by combining structural and content information of all data records in corresponding category. Then, the network parameters are learned from all training data records in this category. To realize fault diagnosis, we classify test log data records into possible fault categories by performing inference in constructed CBNs.

Consider the sample XML log files F0 and F1 in Fig. 1. For the training data in a given category c, the schema information of each log record can be represented by T_0 or T_1. According to this, we can construct the basic structure of CBN by combining T_0 and T_1, which contains a set of structural nodes denoted by Ns. In addition, there are also a set of content nodes (as the leaf nodes in CBN) denoted by Nt for representing textual information of log data. For simplicity, the discussion below will mainly focus on those structural nodes. Fig. 2 shows the basic structure of CBN, which is a directed graph with T_0 and T_1 as its subgraphs. In fact, each node in this CBN has node c for the corresponding category as its father (which is omitted for sake of simplicity). According to quantified similarity results, the structural nodes can be divided into two groups, including combination group and private group. The combination group contains the nodes which have the corresponding nodes or their similar counterparts in both T0 and T1. In comparison with combination group, the private group is composed of the nodes, which only have the corresponding nodes in T_0 (or T_1).

4. Similarity-based Bayesian learning for fault diagnosis

In this section, we propose a similarity-based Bayesian learning approach for fault diagnosis. Firstly, we introduce how to estimate similarity degrees of schema elements for constructing combined Bayesian networks. Then, we present how to learn probabilities o CBNs in detail. Finally, we give the method of classifying log data records using CBNs for fault diagnosis in subsection 4.3. To illustrate this learning approach, we will continue to use the example given in Section 3.

4.1. Estimating similarity degrees for constructing CBNs

Since the probabilities of CBNs depend on the distribution of both corresponding data and their similar counterparts in training dataset, we need to find the correspondences between the nodes of extracted schema trees. Consider the schema trees T_0 and T_1 in previous example. There are a number of nodes in T_0 and T_1 which have similar meaning but different names. In order to discover the correspondences between these nodes, we will exploit both textual and structural information of T_0 and T_1.

The task of finding consistency between similar elements of two schemas is often regarded as matching. In this subsection, we will estimate element similarity degrees by schema matching. As for the matching method, we have made a reference to the similarity flooding algorithm [26]. This algorithm takes two graphs (schemas, catalogs, or other data structures) as input, and produces a mapping between corresponding nodes of the graphs as output. Based on the similarity flooding algorithm, our method tries to find a mapping between corresponding nodes of the schema trees. Moreover, we utilize graded structural information in different stages, which can improve both of initial mapping accuracy and computation efficiency. In the matching

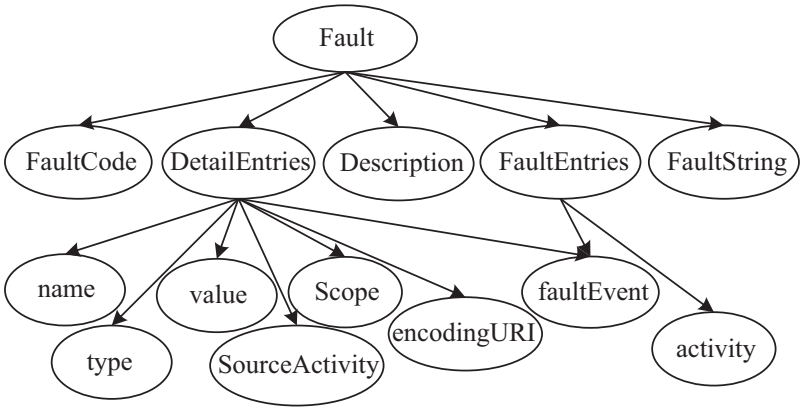


Fig. 2. Basic structure of the CBN by combining T_0 and T_1.

process, we take T_0 and T_1 as input, and produce similarity degrees between corresponding nodes of T_0 and T_1 as output. There are two steps in this process based on pairwise graph, including initial matching and similarity propagation. In initial matching, we additionally exploit the structural information in attribute level. Given two nodes m0 and n1 of T_0 and T_1 which has attribute sets Am and An respectively, we will match attributes of A and A as well as. names of m_0 and n_1. The computation formula is

$$Sim^0(m_0,n_1)=\theta\frac{|A_m\cap A_n|}{|A_m|+|A_n|}+(1-\theta)\ \ StrSim(m_0,n_1) \tag{1}$$

where $|A_m|$ is the number of elements in set A_m, $StrSim(m_0,n_1)$ represents the similarity value of m_0 and n1 by string matching, and is a weight coefficient fixed by users. Based on initial matching results, we construct the pairwise graph in step (2). A portion of this pairwise graph is shown in Fig. 3, where some nodes of low degrees in initial matching are ignored. Then, similarity propagation is executed based on this graph. The computation formula for similarity propagation is

$$Sim^{i+1}(n_0,n_1)=Sim^i(n_0,n_1)+\sum_{(m_0,m_1)\in NeigSet((n_0,n_1))}Sim^i(m_0,m_1)\ \ \omega((m_0,m_1)(n_0,n_1)) \tag{2}$$

where $Sim^i(n_0,n_1)$ indicates the similarity degree of pairwise node (n_0,n_1) in the i-th iteration, $\omega((m_0,m_1)(n_0,n_1))$ denotes the weight of edge between the given nodes (whose value equals to that of 1 divided by the outgoing edge number of the source node for this edge), and $NeigSet((n_0,n_1))$ represents the set of neighbors of node (n_0,n_1) in the pairwise graph.

Table 1
Similarity estimation result.

Node in T_0	Node in T_1	Similarity (Sim)
Fault	Fault	0.90
DetailEntries	DetailEntries	0.68
DetailEntries	FaultEntries	0.44
FaultEvent	FaultEvent	0.19
SourceActivity	Activity	0.19
Name	Name	0.17
Type	Type	0.17
Value	Value	0.17
FaultCode	FaultCode	0.16
Description	FaultString	0.10
Scope	EncodingURI	0.10

This computation process is performed iteratively to estimate similarity degrees between nodes of T_0 and T_1. After this process, the computed similarity degrees will be normalized to the range of [0, 1].

Table 1 shows a portion of the similarity estimation result. If we set the value of filtering threshold to 0.1, the pairs whose similarity degree is no bigger than 0.1 will be deleted from the CBN.

As shown in Table 1, an example of the matching results could be the correspondence between node “DetailEntries” and node ”FaultEntries”.

4.2. Similarity-based Bayesian learning algorithm for computing CBN probabilities

In this subsection, we will introduce how to compute CBN probabilities using similarity-based learning algorithm. According to computed similarity degrees, the conditional probabilities of CBNs will be obtained.

Based on the similarity result, the combination part of CBN can be created as shown in Fig. 4. Then, the conditional probabilities related to the nodes of this part will be learned from training data. In fact, the similar counterparts

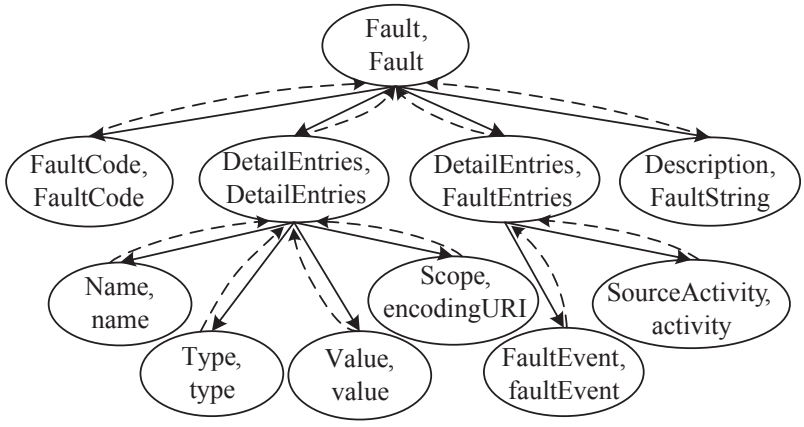


Fig. 3. A portion of pairwise graph for matching T_1 and T_2.

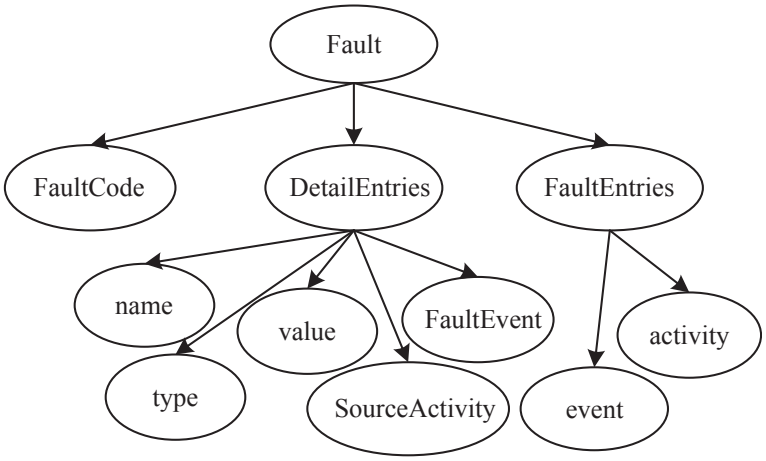


Fig. 4. Combination part of the CBN.

of log records in training dataset also influence corresponding probability distribution of the CBN. Taking the influence of the similar counterparts into consideration, we will calculate the probabilities in the combination part of CBN, based on similarity degrees between nodes of T_i and CBN.

Before going into details of probability computation, it is necessary to introduce the concept of combination similarity degree *CSim*, which represents the similarity between a node of T_i and its corresponding node (with the same name) of constructed CBN. Assume n and s are the CBN nodes which correspond to nodes ni and si respectively. The *CSim* is computed recursively from leaf to root nodes for each schema tree, using the following formula

$$CSim_i(n_i,n)=\frac{1+\sum_{s_i\in SonsOf(n_i),s\in SonsOf(n)} CSim_i(s_i,s)}{1+\max\{|SonsOf(n_i)|,|SonsOf(n)|\}} \tag{3}$$

where $SonsOf(n_i)$ denotes the set of son nodes for node n_i (and it is the same for node n). According to this, we can obtain the combination similarity degrees between nodes of T_0 (T_1) and CBN, as shown in Table 2.

Table 2
Combination similarity degrees.

(a) *CSim* between nodes of T_0 and CBN

Node in T_0	Node in CBN	Combination similarity ($CSim_0$)
Fault	Fault	0.65
DetailEntries	DetailEntries	0.88
FaultCode	FaultCode	1
Description	Description	1
...	...	...

(b) *CSim* between nodes of T_1 and CBN

Node in T_1	Node in CBN	Combination similarity ($CSim_1$)
Fault	Fault	0.77
DetailEntries	DetailEntries	0.63
FaultEntries	FaultEntries	1
FaultCode	FaultCode	1
...	...	...

For the example mentioned above, suppose R_0 and R_1 are the record sets of training data, from which T_0 and T_1 can be extracted. Let R_0^c and R_1^c be the subsets of R_0 and R_1 with fault category c. Then, let R^c be the union of these two subsets, i.e., $R^c = R_0^c \cup R_1^c$. Accordingly, given a log record r_i from record set R_i, we use $SD(r_i, m_i, n)$ to denote the similarity degree of tree node (corresponds to element) m_i of r_i and node n of CBN. The similarity degree *SD* will play an important role in learning process, which can be computed by the following formula

$$SD(r_i,m_i,n)=\begin{cases} CSim_i(m_i,n), & m_i\in CSet(n) \\ Sim(m_i,l)\cdot CSim_{1-i}(l,n), & m_i\in SimSet(l)\wedge l\in CSet(n) \\ 0, & \text{other} \end{cases} \tag{4}$$

where $CSet(n)$ represents the set of tree nodes corresponding to node n (with the same name), and $SimSet(l)$ denotes the set of nodes similar to a tree node l. As an important step in learning, the times of a specific node and its parent appearing in the record set will be counted. Herein, we use $SimNum(r, p, q)$ to represent the similarity-based number of node q having node p as its parent for record r. We compute the value of $SimNum(r, p, q)$ by the formula shown below

$$SimNum(r,p,q)=\begin{cases} \sum_{e,pa(e)\in NodeSet(r)} SD(r,pa(e),p)\ \ SD(r,e,q), & q\in N_s \\ \sum_{pa(e)\in NodeSet(r)} SD(r,pa(e),p)\ \ StrEqua(e,q), & q\in N_t \end{cases} \tag{5}$$

where $pa(e)$ denotes the parent node of node e in record r, $NodeSet(r)$ is the node (or element) set of record r, and $StrEqual(e, q)$ means whether e equals to q by string matching.

For the CBN of category c, we suppose each node has node c as its father. Thus, the root element of each record in this category has the same parent node c. The conditional probability can be computed as follows

$$p(n=q|pa(n)=p,c)=\frac{\sum_{r\in R^C} SimNum(r,p,q)}{\sum_{u\in N_s\cup N_T}\sum_{r\in R^C} SimNum(r,p,U)} \quad (6)$$

Based on above computation formulas, the similarity-based learning algorithm is shown below.

Algorithm Similarity-based Bayesian learning

Input: A training record set $R^c = R_0^c \cup R_1^c$ with category c; A CBN node set $N = N_s \cup N_T$.

Output: $P(q \mid p, c)$

```
for each node q ∈ N do
    for each record r ∈ R^c do
        Calculate SimNum(r, pa(q), q) and add it to numSum;
        for each node u where pa(u) = pa(q) ∧ u ≠ q do
            Calculate SimNum(r, pa(u), u) and add it to denSum;
    P(q | p, c) = numSum / (numSum+denSum);
return P(q | p, c)
```

Given the training records in a category c and the node set of constructed CBN, we can compute the conditional probability $P(q \mid p, c)$ for each node q and its parent p in the CBN, using this learning algorithm.

4.3. Similarity-based Bayesian learning algorithm in dynamical environment

Considering the dynamic nature of log data from realworld systems, we need to update the existing classification model to get accurate diagnosis in dynamic environment. Actually, the dependent information of existing CBN model may be inaccurate or incomplete when training data are changing. It requires dynamically updating the structure and dependency probabilities of the CBN model. To obtain a more accurate generation model, we dynamically update existing CBN model based on incremental training data, by (1) adding new nodes to current node set, (2) changing dependency between nodes, and (3) calculating the dependency probability for the updated dependency set.

Depending on new training data, we incrementally generate the classification model $CBN\Delta$ for different categories. Then, for a specific category c, we dynamically update the probability set P of existing CBN_0, based on the structure G_Δ^c and probability set P_Δ^c of newly constructed $CBN\Delta$. In this subsection, we propose the dynamic similarity-based learning algorithm, which can update the existing model and calculate its corresponding probabilities when new training data are arriving.

In this algorithm, according to the obtained similarity, we calculate the similarity degree SD_Δ for different categories between nodes of new model CBN_Δ and the existing model CBN_0. Herein, $SD_\Delta(m_\Delta, n, c)$ denotes the similarity degree for category c between the node m_Δ of new model CBN_Δ and the node n of current model CBN_0. The corresponding formula is shown as follows.

$$SD_\Delta(m_\Delta,n,c)=\begin{cases} CSim_i(m_i,n), & m_\Delta\in EquSet(m_i)\\ \text{argmax}Sim_{\Delta i}(m_\Delta,m_i)\cdot CSim_i(m_i,n), & m_\Delta\in SimSet(m_i)\\ 0, & \text{other}\end{cases} \quad (7)$$

where $EquSet(m_i)$ and $SimSet(m_i)$ represent the equivalent set and the similar set of a node m_i from schema tree T_i, respectively. Herein, $Sim_{\Delta i}(m_\Delta, m_i)$ denotes the similarity degree between node m_Δ and node m_i. In other words, if node m_Δ is equivalent to a schema tree node m_i, we can obtain SD by directly calculating combination similarity degree of node m_i and node n. Otherwise, when node m_Δ is similar to a schema tree node m_i, we can get SD by multiplying the similarity degrees $Sim_{\Delta i}(m_\Delta, m_i)$ and $CSim_i(m_i, n)$.

Based on the above computation formula, the dynamic similarity-based learning algorithm is shown below. The focus of the improved algorithm is the dynamical update of the dependency probabilities of the CBN model. In this algorithm, if an edge does not belong to the intersection of G^c andG_Δ^c, its probability will be calculated based on the value of the similarity degree SD_Δ.

Algorithm Dynamic similarity-based learning

Input: Structure G_Δ^c and probability set P_Δ^c of newly constructed CBN_Δ with category c; Structure G and probability set P of existing CBN0 with category c.

Output: $P_{new}(q \mid p, c)$

```
if c ∈ ΔC − C then
    G_new^c = G_Δ^c ;
    for each edge (p,q) ∈ G_new^c do
        P_new(q | p, c) = (q | p, c);
else if c ∈ ΔC I C then
    α = |R^c| / (|R^c| + |R_Δ^c|) ;
    G_new^c = G^c ∪ G_Δ^c ;
    for each edge (p,q) ∈ G_new^c do
        if (p,q) ∈ G^c I G_Δ^c then
            TemP(q | p, c) = P(q | p, c);
            TemP_Δ(q | p,c) = P_Δ(q | p,c);
        else if (p,q) ∈ G^c − G_Δ^c then
            TemP(q | p, c) = P(q | p, c));
            TemP_Δ(q | p,c) =
                Σ_{(p',q')∈G_Δ^c} P_Δ(q' | p',c)×SD_Δ(p',p,c)×SD_Δ(q',q,c);
        else if (p,q) ∈ G_Δ^c − G^c then
            TemP(q | p,c) =
                Σ_{(p'',q'')∈G^c} P(q'' | p'',c)×SD_Δ(p,p'',c)×SD_Δ(q,q'',c);
            TemP_Δ(q | p,c) = P_Δ(q | p,c);
        P_new(q | p,c) =
            α×TemP(q | p,c) + (1−α)×TemP_Δ(q | p,c);
return P_new(q | p,c)
```

Using the proposed dynamic similarity-based learning algorithm, we can update the existing CBN model and calculate its corresponding probabilities based on incremental training data in the dynamic environment.

4.4. Fault diagnosis by classifying log data with CBNs

By labeling the training log records from heterogeneous sources with related categories, fault diagnosis can be viewed as a classification problem for log records obtained in runtime. As mentioned above, we can achieve the goal of fault diagnosis by classifying log data using the proposed generative model. According to this, the main approach of fault diagnosis is to first construct the combination part of CBNs by matching schema trees, then compute the probabilities of CBNs from training data based on estimated similarity degrees, and finally classify the newly obtained log data into possible fault categories using generated CBNs.

In classification task, the CBN model plays a key role which is proposed as a generative model based on Bayesian networks. The log records in training dataset with the same category will share the parameters of a CBN. That is to say, there is a set of such parameters for each fault category. The similarity-based probabilities of the CBNs can improve the accuracy of the classification task. Then, the log records in testing dataset can be classified into possible fault categories, by calculating the probability that each category will generate the log data record. Given a category c and a test log record r_{test}, we can estimate the conditional probability by the formula

$$P(r_{test}|c) = \prod_{n_s \in N_s} P(n_s|pa(n_s), c) \prod_{n_t \in N_t} P(n_t|pa(n_t), c) \tag{8}$$

Given the set of predefined categories, our objective is to assign most probable category labels to unlabeled log records, based on the likelihood inference in corresponding CBNs. According to computed conditional probabilities, we will choose the category c_{MAP} which has the maximum posteriori probability value to label the test log record, as shown below.

$$\begin{aligned} c_{MAP} &= \operatorname*{argmax}_{c \in C} P(c) P(r_{test}|c) \\ &= \operatorname*{argmax}_{c \in C} P(c) \prod_{n_s \in N_s} P(n_s|pa(n_s),c) \prod_{n_t \in N_t} P(n_t|pa(n_t),c) \end{aligned} \tag{9}$$

5. Evaluation

To evaluate our approach for fault classification of Web service flows, we have conducted experiment on both real and synthetic log datasets in this section. We compare the classification results of our approach with those of other classifiers for semi-structured documents. Experimental results show our approach outperforms other two approaches in dynamic and heterogeneous environment.

5.1. Experiment setup

The log data used in our experiments are collected from our Web services platform which is supported by ActiveVOS engine [27]. The log data are generated by the monitoring module of execution engine. The raw log data have different levels including debug, information, warning, error, and fatal levels. In preprocessing step, we have implemented a monitoring module to extract fault related log data, which ignores the low level information (e.g. the information in debug level) of raw log data. To simulate the data obtained from heterogeneous sources, we represent a portion of these log data using different XML structure. Then, we get the training dataset by labeling each record of these XML log data. There are 30 Web services running on this platform which is taken as the testbed of our experiment. Since faults in application and middleware levels are the common causes of failures in Web services execution, we inject 95 such faults into running services instances. In addition, we implement a synthetic data generation program to simulate the creation of log data, based on the symptom database [28] for IBM WebSphere Application Server. We have generated 1000 pieces of log records from 11,601 pieces of XML log records in this database. And the training dataset is obtained by labeling each data record according to the value of "symptomtype" attribute.

5.2. Datasets

Herein, training data and test data are selected randomly in accordance with the 90/10 ratio. Then, 50 percent of training data is used as new data. To simulate the update of the training dataset in dynamic environment, new log data are added to the training dataset at a fixed speed. For each dataset, we design 40 simulation cases, which has a corresponding training and test datasets.

Table 3 shows the detailed information of example datasets used in evaluation. The real and synthetic datasets are further divided into two subsets in different structure, respectively. We select 65 percent of data records in training dataset, which have the structure different from those of the other portion. Then, to show the advantage of our approach, we choose only 30 percent of records in testing dataset, whose structure is the same to that of the major portion of training dataset.

There are some advantages to evaluate our approach on both real and synthetic log datasets. On one hand, we use the real dataset to validate the approach in practical situations. On the other hand, the synthetic dataset help us to study the effects of different kinds of structural patterns.

5.3. Evaluation results

For evaluating the performance of corresponding approaches, we define accuracy as the proportion of log records that are correctly assigned to a category. The average accuracy is used as the key metric in this experiment, which is the mean accuracy over all categories of the real and synthetic datasets.

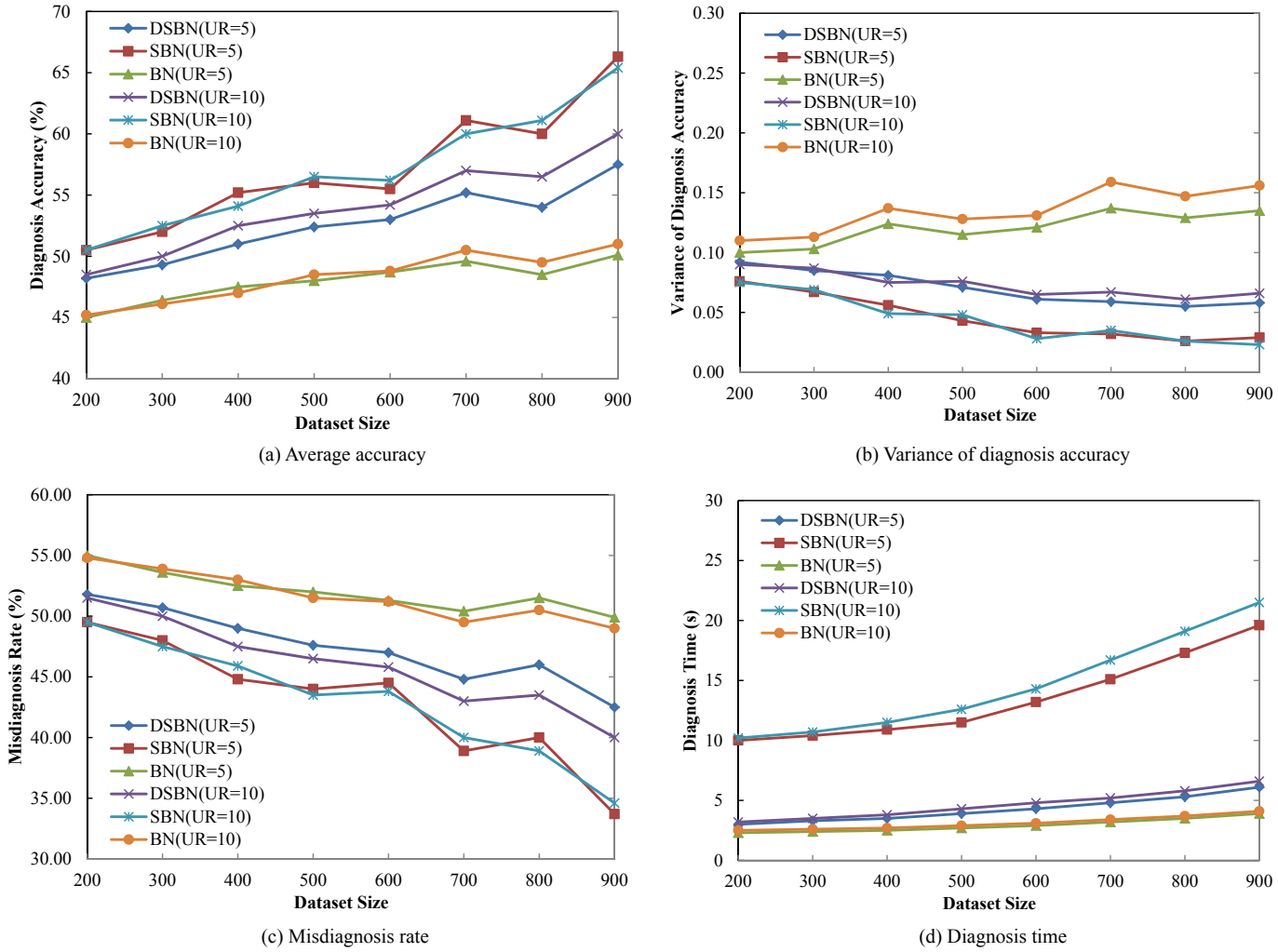
Table 3
Example datasets used in evaluation.

Record number	Real dataset		Synthetic dataset	
	Structure T_0	StructureT_1	Structure T'_0	Structure T'_1
Training Dataset	28	52	315	585
Testing dataset	10	5	70	30

Those algorithms for fault classification are implemented in Java, and runs on a Windows machine with a dual (2.4 GHz) core Intel processor. To show the advantage of our approach in classification task, we compare it with other classifiers designed for semistructured documents, including Bayesian network model (BN).

For simplicity, we use SBN and DSBN to represent original and improved similarity based learning algorithms respectively. Both of SBN and BN methods [22] achieve learning model updates learning from entire training datasets repeatedly. In contrast, DSBN method is able to update its model incrementally in dynamic environment. Considering the dynamic feature, we assume new training data appear at some fixed rate. Herein, we use update rate (UR) to represent the fixed rate and set the value of UR to r records per minute. And the occurrence time of each new record is set randomly. In the simulation of dynamic environment, we add the new training data at various update rates (e.g., $UR = 5$ and $UR = 10$).

In Fig. 5, the comparative results of three classification approaches are demonstrated. Fig. 5a shows comparative result on average diagnosis accuracy of BN, SBN and DSBN algorithms. For different algorithms with the same UR, BN has poor diagnosis accuracy since it cannot handle heterogeneous training data. Though DSBN can efficiently integrate heterogeneous information in dynamic environment, its accuracy has been affected by updating model incrementally. In contrast to DSBN, SBN learns from the entire training dataset from time to time, so it can obtain a precise model and achieve relatively high accuracy. Similarly, due to higher UR, the update speed of training data increases, resulting in the delay of model update. Because of high cost of model update, SBN has downward trends in diagnosis accuracy. Fig. 5b demonstrates the comparative result of accuracy variance of BN, SBN and DSBN algorithms. Considering different algorithms



(a) Average accuracy

(b) Variance of diagnosis accuracy

(c) Misdiagnosis rate

(d) Diagnosis time

Fig. 5. Comparison of diagnosis approaches.

with the same UR, the curves of SBN and DSBN are relatively close to each other. The changing trends of SBN and DSBN seem more stable than BN. For the same algorithm with various UR, higher UR is often companied by larger accuracy variance. In Fig. 5c, we give the comparative result of misdiagnosis rate of BN, SBN and DSBN algorithms. Comparing different algorithms with the same UR, the misdiagnosis rates of SBN and DSBN are lower than that of BN. And SBN has a slight advantage over DSBN in terms of misdiagnosis rate. Since SBN and DSBN can handle heterogeneous data, they have low misdiagnosis rate. Moreover, SBN is able to get a more accurate model by repeatedly learning from entire training dataset, thus it can outperforms DSBN. However, with increasing UR, the misdiagnosis rate of SBN becomes higher than before. Fig. 5d shows the comparative results of diagnosis time. For the same algorithm with different URs, the diagnosis time seems close to each other. While, for different algorithms with the same UR, SBN spends the most time since it need to learn from the whole dataset for updating the generation model. In comparison to SBN, the time cost of DSBN is slightly more than BN, since it only adjusts the generation model in an incremental way. Accordingly, BN is capable of learning from data with the identical structure, so it has the least time consuming.

In terms of diagnosis accuracy and its variance, experimental results show that SBN algorithm outperforms BN algorithm when the size of log dataset is increasing. In contrast, DSBN algorithm has advantages over SBN and BN algorithms, in terms of both diagnosis accuracy and time. Since diagnosis time is a major consideration in fault tolerant service flow execution, *DSBN* algorithm is more suitable for deployment in practical systems, which support diagnosing faults of Web service composition in dynamic and heterogeneous environment.

6. Conclusion

In this paper, we focus on fault diagnosis by analyzing semi-structured log data. By transforming fault diagnosis problem into classification problem, we can utilize the corresponding classification methods to diagnose faults. We propose a similarity-based Bayesian learning approach for constructing combined Bayesian networks, which are used as generative model to classify fault related log data. Different from other approaches, it can learn from training data with different structural information. To realize fault diagnosis, our approach consists of three main steps: (1) estimate similarity degrees of structural elements from different log files, (2) construct the basic structure of CBNs by computing its probabilities using similarity-based learning algorithm, and (3) classify test log data into possible fault categories based on the generated CBNs.

Based on evaluation results, SBN algorithm outperforms BN algorithm when the size of log dataset is increasing. In terms of both diagnosis accuracy and time, the proposed DSBN algorithm has advantages over SBN and BN algorithms. Since diagnosis time is a major consideration in fault tolerant service flow execution, DSBN algorithm should be chosen and deployed in practical system for diagnosing faults of Web service composition.

As the first step, this paper presents our work in fault diagnosis of Web service composition by analyzing log data, especially in dynamic and heterogeneous environment. In the future, we will enlarge the size of log datasets from more real-world service flows. Moreover, we are to study on strategies and mechanisms for optimizing the trade-off between accuracy and efficiency of fault diagnosis for Web service composition.

Acknowledgment

This work is partially supported by National Basic Research Priorities Programme (No. 2013CB329502), National Natural Science Foundation of China (No. 61472468, 61502115), General Research Fund of Hong Kong (No. 417112), and Fundamental Research Funds for the Central Universities (No. 3262014T75, 3262015T20, 3262015T70, 3262016T31).

References

[1] M.P. Papazoglou, P. Traverso, S. Dustdar, F. Leymann, Computer 40 (11) (2007) 38–45.
[2] M.P. Papazoglou, W.-J. Heuvel, VLDB J. 16 (3) (2007) 389–415.
[3] L. Richardson, S. Ruby, RESTful Web Services, O'Reilly, Sebastopol, CA, 2007.
[4] C. Pautasso, O. Zimmermann, F. Leymann, RESTful Web services vs. Big' Web services: making the right architectural decision, in: Proc. International World Wide Web Conference (WWW'08), Beijing, China, 2008.
[5] Q.Z. Sheng, X. Qiao, A.V. Vasilakos, C. Szabo, S. Bourne, et al., Inf. Sci. 280 (2014) 218–238.
[6] A. Alves, A. Arkin, S. Askary, C. Barreto, B. Bloch, et al. (Eds.), Web Service Business Process Execution Language Version 2.0, 2007. Available online at: http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.html.
[7] J. Lee, S.-J. Lee, P.-F. Wang, IEEE Trans. Serv. Comput. 8 (2) (2015).
[8] L. Ardissono, R. Furnari, A. Goy, G. Petrone, M. Segnan, Fault tolerant Web service orchestration by means of diagnosis, in: Proc. European Workshop on Software Architecture (EWSA'06), Nantes, France, 2006.
[9] G. Friedrich, M. Fugini, E. Mussi, B. Pernici, G. Tagni, IEEE Trans. Softw. Eng. 36 (2010) 198–215.
[10] O. Kopp, F. Leymann, D. Wutke, Fault handling in the web service stack, in: Proc. International Conference on Service Oriented Computing, LNCS 6470, 2010, pp. 303–317.
[11] WS-DIAMOND team, WS-DIAMOND: Web Services DIAgnosability, MONitoring and Diagnosis, 2005. Available online at: http://wsdiamond.di.unito.it/.
[12] Y. Yan, P. Dague, Monitoring and diagnosing orchestrated Web service processes, in: Proc. IEEE International Conference on Web Services (ICWS'07), Salt Lake City, Utah, USA, 2007.
[13] Y. Yan, P. Dague, Y. Pencolé, M.O. Cordier, Int. J. Web Serv. Res. (JWSR) 6 (1) (2009) 87–110.
[14] W. Mayer, G. Friedrich, M. Stumptner, Diagnosis of service failures by trace analysis with partial knowledge, in: Proc. International Conference on Service Oriented Computing, LNCS 6470, 2010, pp. 334–349.
[15] Y. Dai, L. Yang, B. Zhang, Z. Zhu, Exception diagnosis for composite service based on error propagation degree, in: Proc. IEEE International Conference on Services Computing (SCC'11), 2011, pp. 160–167.
[16] Z. Zhu, J. Li, Y. Zhao, Z. Li, SCENETester: a testing framework to support fault diagnosis for Web Service Composition, in: Proc. IEEE

International Conference on Computer and Information Technology, 2011, pp. 109−114.
[17] Z. Jia, R. Chen, Hybrid model-based diagnosis of web service compositions, in: Proc. International Joint AAAI Conference on Artificial Intelligence (AAAI'13), Bellevue, Washington, USA, 2013, pp. 1617−1618.
[18] T. Li, F. Liang, S. Ma, W. Peng, An integrated framework on mining logs files for computing system management, in: Proc. ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (SIGKDD'05), Chicago, Illinois, USA, 2005.
[19] W. Peng, T. Li, S. Ma, SIGKDD Explor. 7 (1) (2008) 44−51.
[20] T. Reidemeister, M.A. Munawar, M. Jiang, P.A.S. Ward, Diagnosis of recurrent faults using log files, in: Proc. Conference of the Center for Advanced Studies on Collaborative Research (CASCON'09), Ontario, Canada, 2009.
[21] S. Duan, S. Babu, K. Munagala, Fa: a system for automating failure diagnosis, in: Proc. IEEE International Conference on Data Engineering (ICDE'09), Shanghai, China, 2009.
[22] L. Denoyer, P. Gallinari, Inf. Process. Manag. 40 (5) (2004) 807−827.
[23] M.J. Zaki, C.C. Aggarwal, Xrules: an effective structural classifier for xml data, in: Proc. ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (SIGKDD'03), Washington, DC, USA, 2003.
[24] S. Modafferi, M. Enrico, P. Barbara, SH-BPEL: a self-healing plugin for Ws-BPEL engines, in: Proc. Workshop on Middleware for Service Oriented Computing, Melbourne, Australia, 2006.
[25] X. Han, Z. Shi, W. Niu, K. Chen, X. Yang, Similarity-based Bayesian learning from semi-structured log files for fault diagnosis of Web services, in: Proc. IEEE/WIC/ACM International Conference on Web Intelligence, Toronto, Canada, 2010.
[26] S. Melnik, H. Garcia-Molina, E. Rahm, Similarity flooding: a versatile graph matching algorithm and its application to schema matching, in: Proc. IEEE International Conference on Data Engineering (ICDE'02), San Jose, California, USA, 2002.
[27] ActiveVOS, Available online at http://www.activevos.com/developers/sdks, 2011.
[28] S.K. Chilukuri, K. Doraisamy, Symptom database builder for autonomic computing, in: Proc. International Conference on Autonomic and Autonomous Systems (ICAS'06), Silicon Valley, CA, USA, 2006.