Fault Representation in Case-Based Reasoning

Ha Manh Tran and Jürgen Schönwälder

Computer Science, Jacobs University Bremen, Germany
{h.tran,j.schoenwaelder}@jacobs-university.de

Abstract. Our research aims to assist operators in finding solutions for faults using distributed case-based reasoning. One key operation of the distributed case-based reasoning system is to retrieve similar faults and solutions from various online knowledge sources. In this paper, we propose a multi-vector representation method which employs various *semantic* and *feature* vectors to exploit the characteristics of faults described in semi-structured data. Experiments show that this method performs well in fault retrieval.

Keywords: Case-based Reasoning (CBR), Fault Retrieval, Fault Management, Semantic Search.

1 Introduction

Fault management involves detecting, reporting and solving faults in order to keep the communication networks and distributed systems operating effectively. Managing faults in small and homogeneous networks requires not much effort. However, this task becomes a challenge as networks grow in size and heterogeneity. Proposing solutions for faults not only costs much time and effort but also degrades related network services. Artificial intelligence methods introduce some promising techniques for fault resolution.

The Case-based Reasoning (CBR) [1] approach seeks to find solutions for similar problems by exploiting experience. A CBR system draws inferences about a new problem by comparing the problem to similar problems solved previously. The system either classifies a given problem into a group of already known and already solved problems or proposes new solutions by adapting solutions for related problems to the new circumstance of the problem. Existing CBR systems for fault management usually cooperate with trouble ticket systems to take advantage of the trouble ticket database as the case database. These systems only function on the local case database, and thus limit the capability of exploring fault-solving knowledge present at other sites. Using shared knowledge sources, however, not only provides better opportunities to find solutions but also propagates updates in case databases that otherwise frequently become obsolete in environments where software components and offered services change very dynamically.

Search engines like Google [2] today furnish information search with global data sources and powerful search techniques. It has become common practice for

A. Clemm, L.Z. Granville, and R. Stadler (Eds.): DSOM 2007, LNCS 4785, pp. 50–61, 2007.

people to "google" for a piece of information. Operators are unexceptional; they use fault messages or keywords to search for fault resolution in indexed public archives. Observations have shown that "googling" takes quite some time to find suitable solutions for a given fault. Furthermore, these solutions are typically found in indexed discussion forums, bug tracking and trouble ticket systems, or vendor provided knowledge bases. While some of these data sources maintain some structured information (e.g., bug tracking and trouble ticket systems), this information cannot be exploited due to the usage of a generic search engine which does not understand the meta information readily available.

To deal with this problem, we have proposed a distributed case-based reasoning system [3] which first exploits various fault knowledge sources in a distributed environment to discover similar faults, and then reasons on the retrieved solutions to provide new solutions adapting to the circumstances of new faults. In this paper, we focus on a multi-vector representation method to describe faults as cases in an expressive format, thus allowing the CBR system to retrieve more relevant cases. Intuitively, similar faults can probably be found if they are represented in comparable formats. The paper is organized as follows: in Section 2, we provide some background information about CBR systems and semantics-based search mechanisms. Section 3 explains a novel method to represent faults for case retrieval in the CBR system. The evaluation of this method is presented in Section 4. The related work describes the existing systems in Section 5 before the paper concludes with future work in Section 6.

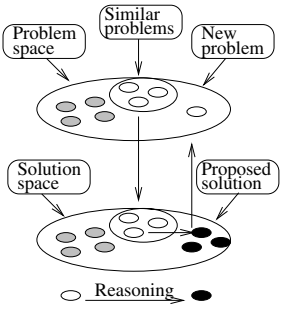

Fig. 1. Case-based reasoning

2 Background

This section provides an overview of the CBR system and the semantics-based search mechanisms (also known as semantic search). A CBR system basically contains four processes: *case retrieval* to obtain similar cases, *case reuse* to infer solutions, *case revision* to confirm solutions and *case retaining* to update new cases. The first process concerns case representation and similarity evaluation; whereas the remaining processes are more related to case reasoning and maintenance. The main focus here is on case representation, similarity functions and semantic search.

2.1 Case Representation and Similarity Functions

A case representation method expresses a case in a formal format to reveal hidden properties and to facilitate case evaluation. Moreover, the representation method also has an influence on the performance of case retrieval. Research in CBR has proposed several representation methods for various domains. The bag of words (BOW) used for law cases tokenizes a text into single words; i.e., a form

of term vectors. The XML-based case representation language (CBML) used for travel markets describes cases in XML format. A review of these methods has been reported in [4]. In addition, other proposals [5,6] in fault management have explored field-value pairs to present faults; i.e., a form of feature vectors. Note that the terms "feature vectors" and "field-value vectors" are used interchangeably throughout this paper.

The structure of trouble tickets [5,6] has been used to represent cases in CBR systems. An example ticket encompasses fields such as "Device name", "Device type", "IP address", "Trouble", "Additional data", "Resolution" plus other fields used for management purposes (e.g., "Owner", "Status", "Time"). Cases are represented as {*field*:*value*} pairs, where *value* is either numeric or symbolic. Case similarity is measured by the number of pairs matched.

Knowledge management systems [7,8] and practical CBR applications [6,9] express cases in *feature* vectors $<f_1{:}v_1,\ldots,f_n{:}v_n>$, where n is number of features; f_i is a domain-specific feature pre-defined in knowledge sources; v_i is a value of the respective feature, which avoids using natural language. This representation not only supports semantic search, but can also be used to represent cases in CBR systems.

Case similarity is calculated by various methods. The *global similarity* method [9,7] takes the significance of features into account:

$$sim(q,c) = \sum_{i=1}^{n} w_i sim(q_i, c_i) \tag{1}$$

n is the number of features; q_i and c_i are features of cases q and c respectively; $sim(q_i, c_i)$ is the distance between q_i and c_i, which are expressed in binary, numeric or symbolic values; w_i is a weight of the i^{th} feature such that $\sum_{i=1}^{n} w_i = 1$ with $w_i \in [0,1]\ \forall i$. A weight is a user-defined value which exhibits the significance of a certain feature.

The *logical match* method [8] uses a logical model to express a case in a set of *predicates* $<f_1{=}v_1,\ldots,f_n{=}v_n>$, where each predicate $\{f_i{=}v_i\}$ is a field-value pair. A case C_i *matches* a case C_j, denoted $C_i \subseteq C_j$, if C_j holds for all predicates in C_i:

$$\forall k \{f_k = v_k\} \in C_i, \{f_k = v_k\} \in C_j \Longrightarrow C_i \subseteq C_j \tag{2}$$

This method supports a *partial match* for the heterogeneous cases that contain different numbers of features.

The *word similarity* method [10,11] compares cases using the hierarchical word structure, namely the taxonomy tree.

$$sim_{Topic}(q,c) = \begin{cases} e^{-\alpha l} \frac{e^{\beta h} - e^{-\beta h}}{e^{\beta h} + e^{-\beta h}} & \text{if } q \neq c \\ 1 & \text{otherwise} \end{cases} \tag{3}$$

l is the length of the shortest path between the topics of q and c in the taxonomy tree, h is the depth level of subsumer in the taxonomy tree, and $\alpha \geq 0$ and $\beta > 0$ are parameters scaling the contribution of shortest path length l and depth h, respectively.

2.2 Semantic Search

Resources and queries in semantic search are expressed in formal formats semantically understandable to search engines. Knowledge management systems describe resources in feature or semantic vectors and evaluate the similarity between vectors using similarity functions. Existing systems have employed different methods including the schema-based method [11,12] for resources related to structured, domain-specific data, and the Latent Semantic Indexing method (LSI) [13] for resources described textually.

Fulltext-Based Search. The LSI method brings the essential abstract concepts of a document or query to a semantic vector. To generate this vector, a document or a query is first represented in a term vector. The Vector Space Model (VSM) [14] weights each term, denoted by $w(t)$, in the term vector by calculating the appearance frequency of this term in the document and the appearance frequency of this term in other documents, as follows:

$$w(t) = \frac{n_{t \in d}}{N_d} \log \frac{N}{n_{d \supset t}} \tag{4}$$

$n_{t \in d}$ is number of term t in document d; N_d is number of terms in document d; N is number of documents; $n_{d \supset t}$ is number of documents containing term t. A high frequency of a term indicates the significance of the term in the document, but its significance is compensated if the term also appears in many other documents. LSI deals with noise and synonyms in a document by transforming a term vector into a semantic vector. To carry out the transformation, LSI represents all documents and terms in the documents in a $t \times d$ matrix A, where each element a_{ij} computed by Eq. 4 denotes the significance of term i in document j. Using singular value decomposition (SVD) [14], A is decomposed into the product of three matrices: $A = U \Sigma V^T$, where $\Sigma = diag(\sigma_1, \ldots, \sigma_r)$ is an $r \times r$ diagonal matrix, r is the rank of A and σ_i is the singular value of A with $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_r$. LSI eliminates noise and synonyms by picking up the s largest singular values resulting in reducing the rank of A; e.g., $A_s = U_s \Sigma_s V_s^T$. The optimal value of s is chosen depending on r; e.g., between 50 and 350. Semantic vectors of documents in A are indexed by the rows of V_s. Semantic vectors of new documents or queries are computed by using U_s, Σ_s [14]. The similarity between two vectors is measured by the cosine of the angle between these vectors. Formally, given two vectors $q = (q_1, q_2, \ldots, q_s)$ and $c = (c_1, c_2, ..., c_s)$ normalized with $\|q\| = 1$ and $\|c\| = 1$, the similarity between q and c is computed by the following equation:

$$cos(q, c) = \sum_{i=1}^{s} q_i c_i \tag{5}$$

Schema-Based Search. The schema-based method (also known as metadata-based or ontology-based search) maps the properties of resources in a pre-defined schema into feature vectors. A schema here denotes the structure of data; e.g., a schema of a digital document consists of title, author, abstract, etc. A query

is also expressed in a feature vector using a schema. Fig. 2 plots the process of the schema-based search. This method works based on knowledge sources that globally define concepts and their relationships related to some domain of interest. These concepts are employed to specify the structured data of resources or queries in schemas. The similarity evaluation of feature vectors has been discussed in 2.1.

2.3 Case Reasoning

While case retrieval is only responsible for producing similar cases, case reasoning deduces from the retrieved cases relevant solutions for the problem, see Fig.1. The deductive capability of case reasoning lies upon an intelligent process named *case adaptation*. This process basically carries out two tasks: the first task distinguishes a retrieved case from the problem to clarify key differences, then the second task modifies the retrieved case following the differences. Instructions from operators take vital roles in these tasks, thus improving the self-adapting capability is the major challenge of case reasoning. Furthermore, case reasoning also undertakes the process of *case retaining* that submits the changes of the adapted cases to the case base after processing case adaptation. It is essential that the process of case learning verifies the results of applying the adapted solutions to a real system before case databases are updated. However, this process is difficult to be performed because real test systems are sometimes unavailable in decentralized environments.

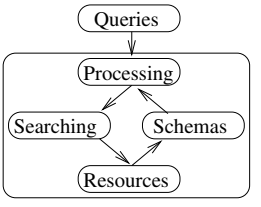

Fig. 2. Schema-based search

3 Fault Retrieval

This section proposes a multi-vector representation method to represent faults. Evaluating the similarity between faults involves several functions corresponding to the represented vectors and an aggregation function to calculate the final similarity value.

3.1 Multi-vector Representation

The heterogeneity of describing cases leads to difficulties in representing cases resulting in vectors with various dimensions and features. The comparison of these vectors thus becomes troublesome and imprecise. A feature vector is either limited by the pre-defined features or composed of the unpredictable user-defined features. It is also difficult to use this vector to explore the properties of the textual cases. On the other hand, a semantic vector exploits the textual cases, but neglects the importance of the case features. This vector only works with the local case database; e.g., two cases retrieved from two different case bases by comparing their vectors are possibly semantically different.

The proposed multi-vector representation method describes cases using a set of vectors instead of a single vector. This method deals with the above problems by breaking a case into various semantic and feature vectors resulting in both expressing cases more effectively and facilitating the comparison of these vectors. In addition, this method is suitable for faults which usually contain hierarchical fields, parameters, textual symptoms. The set of vectors takes advantage of these factors to discover the properties of cases. Nevertheless, introducing several vectors requires an aggregation function to evaluate the similarity between vectors. A network fault is anatomized by the following concerns:

- Field-value pairs classify a fault into the smaller groups of network faults. As described in [6], these groups are related to connectivity, communication performance, authentication service, hardware and software configuration. A case contains several pre-defined field-value pairs such as problem type and area, hardware, platform, and other user-defined field-value pairs. To represent n pairs, we employ the field-value vector: $v_f = <f_1{:}v_1, \ldots, f_k{:}v_k, \ldots, f_n{:}v_n>$, where k is the fixed number of pre-defined pairs.
- Other field-value pairs specify symptoms and typical parameters such as port number, cache buffer, packet loss, error messages depicted in domain-specific terminology. These pairs are represented by another field-value vector $v_p = <p_1{:}v_1, \ldots, p_m{:}v_m>$, where m is number of symptoms and parameters. Symptoms are either binary, numeric or symbolic values. This vector is useful for faults with diagnosis information.
- Textual descriptions including effects, symptoms, debugging message and additional information are represented by the semantic vector v_s using LSI. This high-dimension vector exhumes the properties of a case hidden in the natural language, thus distinguishing the case from other cases. Indexing fault cases and generating query vectors only work with local case databases.

A case, in fact, contains problem and solution parts. A set of vectors $\{v_f, v_p, v_s\}$ can be used to represent the problem part (of a network fault). It is natural to extend the set of vectors to the solution part resulting in more vectors added to the set; the similarity between cases possibly becomes more precise. However, to make the proposed method simple and feasible, we only insist on using the vector set of the problem part for retrieving similar faults; the extended vector set of the solution part related to reasoning the retrieved cases and providing the best case is not discussed in this paper. The following example is a fault extracted from a networking forum [15]:

> *Problem*: Hub connectivity not functioning
> *Description*: The WinXP network contains many machines obtaining an address from a DHCP server. The network is extended to 3 more machines by unplugging the LAN cable from one of the machine's and plugging it to a hub with the intention to add the 3 new machines. From the hub, none of the machines successfully obtains an IP address from the DHCP server; an error message shows "low or no network connectivity".

To make this fault case understandable and comparable to CBR engines, vector v_f contains <problem_type: connectivity, problem_area: hardware configuration, hardware: PC, platform: WinXP>. Vector v_p comprises <network: LAN, error-message: low or no network connectivity, ip-address: false, DHCP: true>. Using LSI, several terms are considered to build the vector v_s.

3.2 Similarity Evaluation

A similarity evaluation function measures the essential properties of cases to conclude the degree of similarity between cases. This function usually depends on the representation method, and therefore has an impact on the performance of case retrieval. For the proposed representation method, the field-value vectors v_f and v_p are evaluated by Ordered Weighted Averaging (OWA) [16,9], which is an aggregation function for multi-criteria decision making, see Eq. 6. This function is suitable for a scenario where the information of the importance of features is unknown, but the order of the importance of features is possibly exploited. It means that the pre-defined features are considered more important than user-defined features, thus receiving higher weight values.

$$sim(q,c) = \sum_{i=1}^{n} w_i sim_{\sigma(i)}(q_i, c_i) \tag{6}$$

where n, q_i and c_i are already discussed in Eq. 1; $sim_{\sigma(i)}(q_i,c_i)$ is a distance between q_i and c_i expressed in binary, numerical or symbolic values. $\sigma(i)$ is a permutation of 1,...,n such that $sim_{\sigma(i)}(q_i, c_i) \geq sim_{\sigma(i+1)}(q_{i+1}, c_{i+1})$ $\forall i = 1, \ldots, n-1$. We compute a weight value w_i using the following function:

$$w_i = \begin{cases} \frac{2}{n+2i} & \text{if } i < \frac{n}{2} \\ \frac{1}{2i} & \text{if } i \geq \frac{n}{2} \end{cases} \tag{7}$$

This monotonic function decreases from $\frac{2}{n+2}$ to $\frac{1}{2n}$, corresponding to the importance of features, as i increase 1 to n; besides, the function guarantees $\sum_{i=1}^{n} w_i \approx 1$. The similarity between semantic vectors v_s is evaluated by the inner product of vectors, see Eq. 5. In summary, given a case c, a query q and the similarity values $sim_{v_f}(q,c)$, $sim_{v_p}(q,c)$, $cos_{v_s}(q,c)$ for the corresponding vectors v_f, f_p, v_s, the similarity $\mathcal{S}$ between c and q is measured by the aggregation function Eq. 1:

$$\mathcal{S}(q,c) = \alpha sim_{v_f}(q,c) + \beta sim_{v_p}(q,c) + \gamma cos_{v_s}(q,c) \tag{8}$$

Parameters α, β and γ specify the significance of vectors provided by users; for instance: $\alpha = 0.4$, $\beta = 0.2$ and $\gamma = 0.4$.

4 Evaluation of Multi-vector Representation

The goal of this evaluation is to show the performance of the multi-vector representation method in terms of retrieving relevant documents. We focus on two

methods: (1) the LSI method using the single semantic vector (*lsi* for short), and (2) the combined method using the two semantic and field-value vectors (*lsi+fvv* for short). We have used the CISI and MED bibliographic datasets [17] with 1460 and 1033 titles and abstracts (*documents* for short) respectively. These datasets provides the textual queries and the corresponding numbers of relevant documents for evaluating document retrieval; besides, the keyword-based queries are also included, as the following example:

> *A textual query*: How can actually pertinent data, as opposed to references or entire articles themselves, be retrieved automatically in response to information requests?
> *A keyword-based query*: *and* (*or* ("data", "information"), *or* ("pertinent", "retrieved", "requests", "automatically", "response", *not* (*or* ("articles", "references"))));

These two queries are the same, but the keyword-based query specifies main keywords and their importance using operators: *and*, *or*, *not*. Therefore, the query could be employed as a field-value vector by assigning different weight values to keywords. Each term possesses a weight value of $\frac{1}{\eta\theta}$, where η is number of *or* groups, θ is number of terms in the group containing the term. Terms in *not* groups receive negative values. For the above example, "data", "pertinent" and "articles" possesses weight values of 0.25, 0.08 and -0.04 respectively. Sum of weight values for a query is 1 except for queries with *not* operators.

The core component of *lsi* is to compute SVD for the large term-document matrix built on the dataset. We have implemented the Jacobi algorithm [18] for computing SVD. The advantage of this algorithm is high accuracy, which is very crucial for resolving this large, sparse matrix with small elements. The issue of *lsi+fvv* is to determine the importance of specific keywords similar to features in field-value vectors; we simply use the same method as Eq. 4 for documents and the additional operators: *and*, *or*, *not* for queries. We experimentally choose $\alpha = 0.4$ for semantic vectors and $\beta = 0.6$ for field-value vectors in the aggregation function Eq. 8; i.e., given a query, the similarity value for each document is aggregated by cosine and global similarity values.

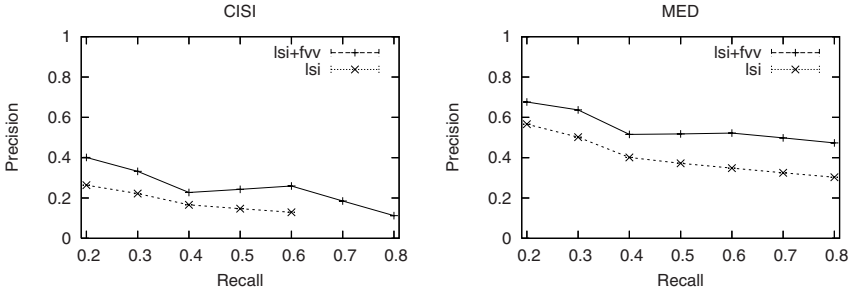

Fig. 3. Precision by various recall rates for the CISI and MED datasets

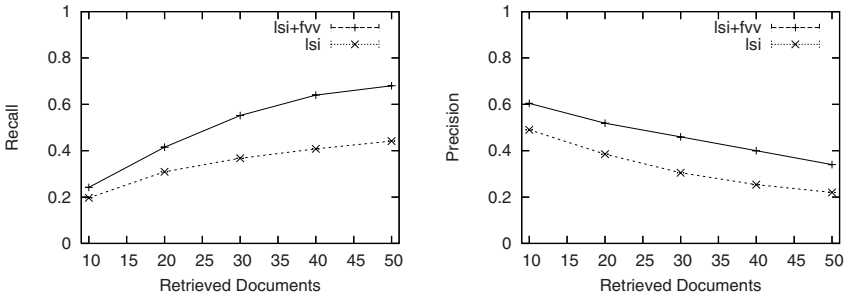

Fig. 4. Recall and precision by various numbers of retrieved documents for the MED dataset

We have considered two popular metrics to evaluate the performance of document retrieval in the experiments: *recall rate* (r_c) and *precision rate* (r_p) [13]. The recall rate is the ratio of the number of relevant documents retrieved to the pre-defined number of relevant documents. The precision rate is the ratio of the number of relevant documents retrieved to the total number of documents retrieved (R_d). Intuitively, the former only concerns the capability of obtaining many relevant documents regardless R_d; whereas, the later involves the capability of achieving relevant documents in the limited number of documents retrieved. These two rates are usually opposed as R_d increases.

We have used 40 queries for CISI and 30 queries for MED. The first experiment evaluates the precision of *lsi+fvv* and *lsi* over the recall rate; i.e., we keep retrieving documents until recall rates are reached, then count the number of documents retrieved and compute precision rates. Fig. 3 shows that r_p decreases as r_c (and also R_d) increases. In the CISI plot, both methods perform poorly for CISI; *lsi* cannot go over the recall rate of 0.6 because the similarity values of retrieved documents go down below 0; *lsi+fvv* performs slightly better. We found the same performance of *lsi* for CISI in [13]. According to this paper, the homogeneous distribution of CISI and the vague description of queries cause the unreliable judgment of the evaluation function. In the MED plot, both methods perform better for MED; *lsi+fvv* slowly reduces the precision rate and remains 0.48 as the recall rate reaches 0.8; whereas, *lsi* acquires 0.3 at the recall rate of 0.8, which is relatively low compared to the precision of *lsi+fvv*. An observation shows that choosing the size of semantic vectors influences the precision of *lsi*; i.e., the reduced rank of the matrix, and choosing the α and β values affects the precision of *lsi+fvv*.

Since the MED dataset provides more reliable results, the second experiment uses this dataset to calculate the accumulative recall and precision rates for different numbers of retrieved documents; i.e., 10, 20, ..., 50. Fig. 4 indicates that *lsi+fvv* outperforms *lsi* in both recall and precision. An observation shows that several relevant documents could be obtained by *lsi+fvv*, while they possess low similarity values in *lsi*; thus, the field-value vector plays a vital role in *lsi+fvv*. Another observation is that *lsi* tends to be misled by queries with *not* operators,

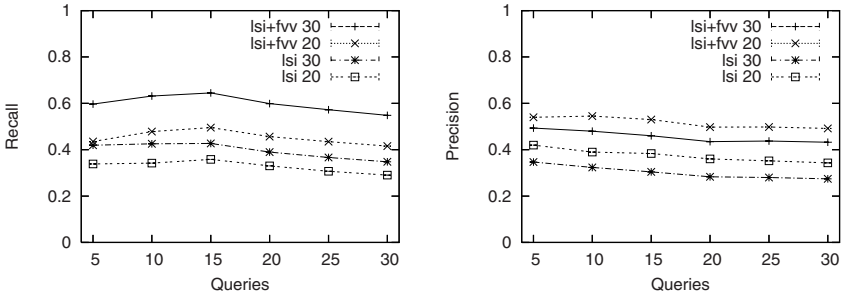

Fig. 5. Recall and precision by various numbers of queries for 20 and 30 documents of the MED dataset

while *lsi+fvv* tends to perform well for queries with distinguished keywords. The precision rate quickly reduces because some queries possess the small number of relevant documents compared to 40 or 50 retrieved documents; for example, a query with 20 relevant documents acquires 16 relevant documents for 40 retrieved documents and 17 relevant documents for 50 retrieved documents, its precision reduces from 0.4 (16/40) to 0.34 (17/50).

To further investigate these methods, we choose 20 and 30 retrieved documents to observe how they perform since the average number of relevant documents per a query is 23.3. Fig. 5 also displays the accumulative recall and precision rates for different numbers of queries; i.e., 5, 10, ..., 30. These rates are relatively stable, thus all queries acquire the similar ratios of relevant documents to the number of retrieved documents (precision) and to the pre-defined number of relevant documents (recall).

5 Related Work

So far several attempts have been given to CBR in fault management. Most of the existing approaches focus on using trouble tickets to represent cases. The work in [5] improves fault management by extending the trouble ticket system (TTS) to CBR. The proposed system can learn from previous experience and offer solutions to novel faults. This work employs trouble tickets as cases for CBR. Likewise, the DUMBO system [6] takes advantage of the knowledge hoarded in TTS to propose solutions for problems. The system not only contains six types of features to express cases but also provides both similarity and reliability measurements for evaluating features between cases. However, these systems are relatively limited by two aspects: (1) the representation of trouble tickets is only suitable for simple feature matching mechanisms, thus restricting the exploitation of features; (2) the knowledge source is limited by using local case databases. Another interesting work [9] uses sets of pre-defined case attributes and preferences of users to improve case representation in recommendation systems. This paper involves using the multi-vector representation and

the advanced similarity evaluation to improve fault retrieval, which is the core component of the proposed distributed CBR system.

Other research activities have concentrated on knowledge management systems working on both multiple case databases and semantic search on peer-to-peer (P2P). Shlomo et al. [7] proposes an approach to retrieving cases on a structured P2P network with the hypercube topology. The approach employs the schema-based method, namely unspecified ontology, to maintain the case base on P2P environment. Case retrieval is based on the approximated search algorithm for feature vectors only, and the focal domain is e-commerce advertisement. The Bibster or SWAP system [11] supports bibliographic data storage and ontology-based search on a super-peer network. The Piazza system [8] deals with the problem of sharing semantically heterogeneous data on a P2P network. These two systems also use the schema-based method to define shared data and retrieve data by evaluating the similarity between feature vectors. The proposed distributed CBR system associates the semantics-based search mechanism with CBR to support not only fault retrieval but also fault-solving capability.

6 Conclusion

Our research aims at building a distributed CBR system to assist operators in finding solutions for faults. The system is more relevant than general search engines because it enables not only searching for similar faults described in semi-structured data but also producing new solutions for new faults. In this paper, we address the problem of retrieving similar faults in the CBR system. By studying the description of fault cases, we propose a multi-vector representation method which uses several feature and semantic vectors to express faults. These vectors not only exploit better the characteristics of faults described in semi-structured data but also provide facilities for evaluating the similarity between faults, thus ameliorating fault retrieval.

We have tested the performance of the proposed method using the CISI and MED bibliographic datasets whose documents contain semi-structured data. The evaluation results show that the combination of semantic and feature vectors outperforms the use of single semantic vectors in terms of document retrieval. Future work will focus on using real fault datasets whose diversity may demand various vectors instead of two vectors. In addition, the proposed method will be extended to case reasoning, which infers the best case from the vector sets of the retrieved cases.

Acknowledgement

The work reported in this paper is supported by the EC IST-EMANICS Network of Excellence (#26854).

References

1. Aamodt, A., Plaza, E.: Case-based reasoning: foundational issues, methodological variations, and system approaches. AI Communications 7(1), 39–59 (1994)
2. Google search engine. Accessed in (May 2007), http://www.google.com/
3. Tran, H.M., Schönwälder, J.: Distributed case-based reasoning for fault management. In: Proc. 1st Conference on Autonomous Infrastructure, Management and Security, pp. 200–203. Springer, Heidelberg (2007)
4. Weber, R.O., Ashley, K.D., Brüninghaus, S.: Textual case-based reasoning. The Knowledge Engineering Review 20(3), 255–260 (2005)
5. Lewis, L.M.: A case-based reasoning approach to the resolution of faults in communication networks. In: Proc. 3rd International Symposium on Integrated Network Management (IFIP TC6/WG6.6), North-Holland, pp. 671–682 (1993)
6. Melchiors, C., Tarouco, L.M.R.: Fault management in computer networks using case-based reasoning: DUMBO system. In: Proc. 3rd International Conference on Case-Based Reasoning and Development, pp. 510–524. Springer, Heidelberg (1999)
7. Berkovsky, S., Kuflik, T., Ricci, F.: P2P case retrieval with an unspecified ontology. In: Proc. 6th International Conference on Case-Based Reasoning, pp. 91–105. Springer, Heidelberg (2005)
8. Tatarinov, I., Ives, Z., Madhavan, J., Halevy, A., Suciu, D., Dalvi, N., Dong, X., Kadiyska, Y., Miklau, G., Mork, P.: The piazza peer data management project. SIGMOD Record 32(3), 47–52 (2003)
9. Montaner, M., López, B., de la Rosa, J.L.: Improving case representation and case base maintenance in recommender agents. In: Proc. 6th European Conference on Advances in Case-Based Reasoning, pp. 234–248. Springer, Heidelberg (2002)
10. Li, Y., Bandar, Z.A., McLean, D.: An approach for measuring semantic similarity between words using multiple information sources. IEEE Transactions on Knowledge and Data Engineering 15(4), 871–882 (2003)
11. Haase, P., Broekstra, J., Ehrig, M., Menken, M., Mika, P., Plechawski, M., Pyszlak, P., Schnizler, B., Siebes, R., Staab, S., Tempich, C.: Bibster — a semantics-based bibliographic peer-to-peer system. In: Proc. 3rd International Semantic Web Conference, pp. 122–136. Springer, Heidelberg (2004)
12. Nejdl, W., Wolf, B., Qu, C., Decker, S., Sintek, M., Naeve, A., Nilsson, M., Palmér, M., Risch, T.: EDUTELLA: a P2P networking infrastructure based on RDF. In: Proc. 11th International Conference on World Wide Web, pp. 604–615. ACM Press, New York (2002)
13. Deerwester, S., Dumais, S., Landauer, T., Furnas, G., Harshman, R.: Indexing by Latent Semantic Analysis. JASIST 41(6), 391–407 (1990)
14. Berry, M.W., Drmac, Z., Jessup, E.R.: Matrices, vector spaces, and information retrieval. SIAM Review 41(2), 335–362 (1999)
15. Networking forum. Accessed in (March 2007), http://www.computing.net/
16. Yager, R.R.: On ordered weighted averaging aggregation operators in multi-criteria decision making. IEEE Transactions on SMC 18(1), 183–190 (1988)
17. Latent semantic indexing. Accessed in (March 2007), http://www.cs.utk.edu/~lsi/
18. Demmel, J.W.: Applied numerical linear algebra. In: Society for Industrial and Applied Mathematics, Philadelphia, PA, USA (1997)