A Case for Usage of Case Usage in Case-Based Reasoning

Mohamed A.K. Sadiq and Deepak Khemani

Department of Computer Science and Engineering,
Indian Institute of Technology Madras,
Chennai – 600 036, India.
sadiq@cs.iitm.ernet.in, khemani@iitm.ac.in

Abstract

Some of the issues in case retrieval and maintenance of case-bases are discussed. Conventionally, nearest neighbor algorithm is applied for retrieval of similar cases. Uniqueness of feature-values is not rewarded in such similarity matching functions. In order to refine the retrieval process, a metric based on successful case usage in the past is incorporated. This metric is shown to be useful in case-base maintenance. Applications in Homoeopathy remedy and personal tour selection are presented with comparative results.

Introduction

The quality of a case-based reasoning (CBR) system depends on the quality of its matching and ranking process (Kolodner, 1993). Matching is the process of comparing the query with stored cases to determine their degree of similarity. Ranking is the process of ordering partially matching cases according to the closeness of match or usefulness.

A classical approach for retrieving similar cases is nearest neighbor algorithm (Duda and Hart, 1973). This method relies on obtaining Euclidean distance kind of measure between the input query and a stored case. The queries as well as the stored cases are modeled as points in a feature space. Based on similarity metrics, distances can be measured between the new problem and the existing cases in this space. It is necessary to assign weights denoting importance of a feature over another while computing the overall similarity between cases. A threshold function filters out the cases, which do not lie within a tolerable distance. It is shown that similarity function must incorporate weights based on distinctive factor of values. Also, the retrieved set of cases is filtered and re-ranked according to the cases' past usage record. The experiments carried out show that these techniques improve the performance of retrieval.

As the number of cases increases, a major issue is in maintaining the large case-base. Case-base maintenance (CBM) is concerned with how addition, deletion and updation of cases take place (Leake, 1996). Case coverage and competence is an increasingly important area where the attempt is to obtain maximal problem coverage with minimal cases and maintenance.

A generic method adopted for case authoring is presented. The application areas are described briefly. Issues related to case retrieval are discussed along with a demonstration of new metrics for retrieval. Then, we describe the use of case usage for refining retrieval and present the evaluation procedure with results. Finally a case-base maintenance approach is proposed.

Case Engineering

The first step in building a case-base is collecting raw case information from source documents within a domain. A case structure is designed and seed cases are collected. Planning is also required for appending new cases in future. A methodology for building a case-base helps in faster knowledge acquisition thereby aiding better case authoring.

Case Acquisition Methodology

The steps involved in the knowledge acquisition process are as follows:

1. Delimit the information regarding a single problem and its corresponding solution from the text. Remove unnecessary data pertaining to other problems.
2. The problem part is identified by the key-phrases.
   2.1. List the features describing the problem.
   2.2. Assign weights to the features, denoting relative importance.
   2.3. Collect the relevant values.
3. The solution part is identified by the diagnosis or actions taken to solve the above problem.
4. If same case already exists in the case-base, discard new case. If problem part alone is same as in a stored case, update the solution part to include new solution. A new case is created if the problem parts are different, although the solution part is same as some stored case. This step is especially useful while retaining new cases in future.

5. Check if the above entries are correct. Verification is done by a domain expert.
6. Name the case and assign a unique case-id. Include comments like the source (from where the case was collected), names of the author (knowledge engineer) and the expert (who verified).

Case Representation

The case-base is represented in a structured form. Every case records the relevant feature-value pairs describing a problem and its solution. The values are either atomic or non-atomic. The non-atomic values include optional qualifiers like adjectives. For instance, a query may have the same adjective-noun pair as in a stored case. Such a case is rewarded higher than a case matching only the noun part. If a query does not have any qualifier, the cases having qualifiers are given less weightage There is no match for solitary qualifiers in the query. Thus, partial matching of non-atomic values is made possible.

Application Areas

The domain of homoeopathy was used for demonstration of this system. All the drugs ranging from Abrotanum to Zincum in Materia Medica (Balakrishnan 2000) were represented as solutions in different cases. The seed case-base was populated with 122 cases. Later 237 real clinical cases were appended to the case-base. Each case has a set of values describing the symptoms segregated into six features viz., location, sensation, concomitants, modalities, cause and other symptoms. Feature weights are assigned by an expert.

Another domain used was that of personal travel agents having 1470 cases. Originally this data set was collected by Lenz for experiments in CBR (Lenz, 1993). The features in this case-base are holiday type, duration, price, number of persons, region, season, transportation, and accommodation. In this domain, the weights are user-defined based on their preferences.

Case Retrieval

The nearest neighbor algorithm is a popular method for case retrieval in domains where linearity is presumed. We show that, it is not sufficient if one merely finds a closest problem in the stored cases. We need to consider how significant is the value in a problem space with respect to its solution space. A *significance* measure is computed based on the number of cases sharing a feature-value and the same solution, and the number of cases having the same feature-value. Only those cases having the same number of feature-value pairs are counted here. *Significance* between a feature-value v_i and a case c_j is defined as follows:

$$S_{i,j} = \frac{|c_{i,j}|}{|v_i|} \qquad (1)$$

where $|c_{i,j}|$ is the number of cases having the feature-value v_i and the same solution as in case c_j

$|v_i|$ is the number of cases having the feature-value v_i with any solution

Always, $|c_{i,j}| \leq |v_i|$. Based on the equation (1), $S_{i,j}$ is a fraction that ranges between 1 / |CB| and 1, where |CB| is total number of cases in the case-base. If a feature-value occurs in many cases, its *significance* is less. Thus, rare values are given more *significance*. If each case has a unique solution, $|c_{i,j}| = 1$ for all cases. However, the same feature-value v_i may occur in *m* different cases. Thus, *significance*, $S = 1/m$ (reciprocal of the number of cases having the value).

The *significance* factor is combined along with the usual weighted average of feature weights. *Importance* of a feature-value is defined as follows:

$$I_j = \frac{w_j}{\sum w} \qquad (2)$$

where w_j is the weight of the feature j (whose value matches a query element)

$\sum_{i=1}^{n} w_i$ (mentioned as $\sum w$ for simplicity) is the sum of weights for the n feature-value pairs relevant to the same solution.

This is a domain specific component. If all the feature-values are equally important for a solution, each feature-value has *importance*, $I = 1/n$, where n is the total number of feature-value pairs that lead to the same solution. The range of *I* is between 0 and 1.

Importance and *significance* are independent of each other, full *importance* does not imply full *significance* and vice versa. Both these factors (*I* and *S*) contribute in determining the relevance between the queried feature-value and a case. Hence, the similarity metric, σ is defined as an average of the *importance* and *significance* factors. The similarity between a queried feature-value v_i and a case c_j is given in equation (3).

$$\sigma_{i,j} = \frac{|c_{i,j}| * \sum w + |v_i| * w_j}{2 * |v_i| * \sum w} \qquad (3)$$

The parameters in the above equation are based on equations (1) and (2). A cumulative score is computed based on this function for every matching query value in the case. Thus, each case is awarded a score depending on its degree of match. As a result, the algorithm retrieves a set of significant neighbors. This approach is termed the Nearest Significant Neighbor (NSN) method.

Related Work

Though the NSN method is unique for CBR, a similar idea is employed in Information Retrieval (IR). Term frequency (*tf*) multiplied by inverse document frequency (*idf*) is predominantly used in Information Retrieval (Salton, 1989). *idf* is a function that assigns a high weight to terms which are encountered in only a small number of documents in the collection. It is supposed that rare terms have high discrimination value and the presence of such a term in both a document and a query is a good indication that the document is relevant to the query.

IR techniques have been applied for retrieval based on *tf* **idf* (Bradshaw *et. al*., 2000), (Budzik, *et. al*., 2001), (Lapalme and Kosseim 2003), apart from other applications (Leake, *et. al*., 2003), (Mock, 1996). We compare the use of such a technique with the similarity metric in equation (3).

The weight of a document given by *tf* **idf* is defined as follows: *tf* is the frequency of term in the document divided by frequency of most frequent word (Baeza-Yates and Ribeiro-Neto, 1999).

$$idf = \log_{10} \frac{|N|}{|t|} \qquad (4)$$

where $|N|$ is the total number of documents in the collection

$|t|$ is the number of documents with the term

Comparison

In CBR systems, similarity function based on *tf***idf* is not suitable for the following reasons:

- *idf* becomes zero when a value (term) occurs in almost all cases (documents). If such a value is queried, the cases would not be weighted at all.
- A maximum value of *idf* can not be determined. It can be generalized as follows: If the term occurs only once in 10^x documents, *idf* = x, where x is any positive number. There is no fixed range.
- Depending on the total number of documents in the collection, *idf* changes. Size of the case-base is not a criterion for matching.
- *tf* is meaningless in CBR because a case will not have repetitive terms like documents.

Using such IR techniques, it becomes very difficult to search for the string "to be or not to be" (Raghavan, 1997). An issue is that the retrieved set of documents is the same when querying with or without the stop-words. For example, Google™ search produces the same result for "not" and for "to be or not to be" (without quotes). This is due to the fact that it considers these words except "not" as stop-words. Query elements could be independent of each other. Hence, co-occurrence is not considered.

Consider a generic example: Let $Q_1=\{v_1\}$ and $Q_2=\{v_1,v_2\}$ be two queries. The query elements v_1 and v_2 are present in the case-base. Feature-value v_1 is present only in some cases whereas, v_2 is in all the cases. The following are assumed without loss of generality:

- Number of cases in the case-base is 200.
- v_1 occurs along with 5 other values in 5 cases termed class A (these cases have 6 feature-value pairs including v_1 and v_2).
- v_1 occurs along with 3 other values in 10 different cases termed class B (these cases have 4 feature-value pairs).
- All cases have unique solutions.
- All the feature-values are equally important (every feature weight, w=1).

Using *idf*, weight of v_2 will be zero. The resultant set will be same for Q_1 and Q_2. Similarity function between a value and a case is given by equation (3) and denoted by σ (value, case). A case in class A is denoted by C_A and that in class B is denoted by C_B.

For Q_1: $\sigma(v_1, C_A) = 0.183$
$\sigma(v_1, C_B) = 0.175$

For Q_2: $\sigma(v_1, C_A) = 0.183$
$\sigma(v_2, C_A) = 0.086$
Cases in class A get a cumulative score of 0.269
For Q_2: $\sigma(v_1, C_B) = 0.175$
$\sigma(v_2, C_B) = 0.128$
Cases in class B get a cumulative score of 0.303

It is noted that Q_1 results in ranking cases in class A higher than those in class B. Whereas, Q_2 ranks cases in class B higher than those in class A. This shows that there is a difference between the results of Q_1 and Q_2. Even if the total number of cases is increased, the above variation is noted. Hence, v_2 should not be ignored as a stop-word.

In order to overcome the issues in *idf*, a *significance* measure is used as described in equation (2). The weighted average of *importance* is used in lieu of *tf*. Thus, the similarity metric given in equation (3) is applied.

Use of Case Usage

CBR assumes that problems close together in feature space are similar to each other. However, some of these problems might have been unsuccessful previously. Useful cases must be given higher ranking. This approach relies on relevance feedback from users. A record is maintained about the usage of case-base and individual cases. Information regarding the case-base includes the number of times the case-base was queried, number of times all the retrieved cases were relevant and number of times all the retrieved cases were irrelevant. Information

about a case counts the number of times a case was retrieved and the number of times it was successful or marked relevant.

Since the usage parameters are independent of the feature-value pairs, their weights and solutions in the cases, this information is not used as part of similarity computation. Once, a set of cases is retrieved (based on NSN method presented in Section 4) these cases will be rewarded or punished depending on whether the cases were used successfully or not in the past. Success of a case is defined as a fraction given in equation (5). This is a local measure of case usage.

$$Success = \frac{c}{r} \qquad (5)$$

where c is number of times the case was marked relevant

r is number of times the case was retrieved

'c' is an integer ranging from 0 to 'r' and $r \geq 1$, since we measure only if the case is retrieved. Thus, the above factor indicates how many times the case was successful out of the times it was retrieved.

Failure rate of a case is defined in equation (6). This is viewed as a global measure of case usage.

$$Failure = \frac{r - c}{t} \qquad (6)$$

where r and c are the same as in equation (5),

t is number of times the case-base was queried.

This factor indicates how many times the case was retrieved and was not marked relevant (unsuccessful retrieval) out of the count of case-base usage (t). The following inequality is true always: $c \leq r \leq t$. Due to this, the failure factor achieves a peak of +1, if the case was always retrieved (whenever the case-base was queried) and never marked relevant (c=0 and r=t). Such cases could be termed as noise. The failure factor becomes zero, if the case is marked relevant whenever it is retrieved (r=c).

Alternatively, if global success rate is required based on the failure rate, it is subtracted from 1. In order to reward or punish the cases depending on their past success or failure, we combine the above factors by summation in equation (7). Thus, usage is defined as follows:

$$U = \frac{t(r + c) - r(r - c)}{rt} \qquad (7)$$

Range of U is [0,2]. U becomes 0 when the case is always retrieved and marked irrelevant (such a case will be dismissed). Although a rare situation, U becomes 1, if t = r = 2*c, that is, the case was always retrieved and half the time it was found relevant. U becomes 2 when the case is always marked relevant (r=c).

A case is deemed to be successful, if it scores $U > 1$, otherwise it is punished for U being less than 1. In order to refine the retrieval based on successful usage of cases, U is multiplied with the case score (similarity with query). Thus, we reward cases with $1 < U \leq 2$ and penalize cases with $0 \leq U < 1$. This approach is termed the Nearest Useful Neighbor (NUN) method.

Conversely, in domains where users would prefer sparingly used cases, the score is multiplied by $(2 - U)$ for rewarding or punishing cases. Thus, the system responds by displaying an ordered set of closely matching potentially useful cases.

Evaluation

An expert homoeopath formulated a set of 25 real-world queries. For every query, a set of actually relevant drugs was obtained independently. This is termed as the correct set, C. The same queries were presented to the system. For each query a set of cases (solution: drugs) was retrieved termed R. Hence, the correctly retrieved drugs are in $C \cap R$. Precision is a measure denoting how many of the retrieved cases were correct. Recall is a measure denoting how many of the correct cases were retrieved. These are given in equation (8) in terms of C and R.

$$Precision = \frac{|C \cap R|}{|R|}, \quad Recall = \frac{|C \cap R|}{|C|} \qquad (8)$$

In the travel agents domain, an anonymous person (who was unaware of the process of case retrieval in the system) posed a set of 25 test queries. Each query was formulated by picking a set of common values from different cases. Ideally these cases (termed correct set) should be retrieved. Precision and recall were used to measure the accuracy of retrieval.

Experiments were conducted on the two case-bases adopting the following four methods independently:

1. nearest neighbor (kNN),
2. IR technique (based on *idf*),
3. nearest significant neighbor (NSN),
4. nearest useful neighbor (NUN).

Precision and recall were measured for every query result using each of these methods. The average values are shown in table 1.

No.	Retrieval Method	Homoeopathy case-base		Travel agents case-base	
		Average Precision	Average Recall	Average Precision	Average Recall
1	k Nearest Neighbor	0.28	0.49	0.46	0.47
2	IR technique (based on *idf*)	0.34	0.59	0.56	0.61
3	Nearest Significant Neighbor	0.42	0.74	0.63	0.69
4	Nearest Useful Neighbor	0.47	0.79	0.73	0.80

Table 1. Comparison of precision and recall for different methods of retrieval

Thus the quality of the retrieval using NUN method is better than that using other methods. *Significance* and *importance* are measured for every query value. Usage is computed for the retrieved cases. Hence, the cases are ranked based on their final scores.

Learning

New cases are appended to the case-base, if they are found to be distinct and consistent with respect to the existing case-base. The process of acquiring cases follows the case authoring method in Section 2. In the medical domain, the cases are obtained from books and clinical records. New cases could be based on queries if distinct from the stored cases. Distinctness is exhibited when a query produces poor result. The best matching cases score low. If the presented solution is acceptable to the new problem, the query elements form a new case which is subject to verification for consistency. In case the solution is not acceptable, an expert suggests a new solution to this query. This is added as a new case with all the query elements along with their feature weights.

Case-Base Maintenance

Over a period of time as the case-base grows, maintenance becomes essential. Without CBM, the case-base will be populated monotonically. Due to this two problems arise predominantly. (a) Retrieval time increases proportionate to the growing number of cases. (b) Inappropriateness of matched set due to retrieval of inapplicable or useless cases that failed to be relevant many times previously.

In order to tackle the first issue, new cases must be appended to the case-base only if they are found to be distinct and relevant to the system. Distinctness is measured in terms of the degree of match between the new case and the closest matching stored case. Relevance is judged by the domain expert. For the second issue of inappropriate retrieval, removal of cases depending on their usage pattern is proposed. Based on the statistics of past usage of case with respect to that of case-base, cases are either termed useful or inapplicable. Cases that are of very less usage are moved from the active case-base to a case archive where sparingly used cases are stored. These cases are not used for retrieval unless the user requests.

Maintenance occurs under any of the following preconditions:

- The case-base is large, for example |CB| >2000
- It has been used several times, e.g., t > |CB|

Each case is examined and archived if necessary. Sparingly used cases are identified depending on the number of times it was retrieved with respect to the case-base usage. For this purpose, r / t is computed for all cases. The cases having below average usage are selected based on the following:

$$\frac{r}{t} < \frac{\sum \frac{r}{t}}{|CB|}$$

In such cases, c / r is computed. Cases are moved to the case archive, if c / r is close to zero and the case has been used to some extent i.e., $r > t / r$. The case archive is reviewed by a domain expert. Thus, useless cases can be deleted.

Apart from recording the number of times the case-base was queried, we also record the number of times all the retrieved cases were relevant (p) and the number of times the relevant set was null (f). Best case-base will have $p / t = 1$, worst case-base will have $f / t = 1$. These two factors are combined together in equation (9). Hit rate is defined as follows:

$$H = \frac{t + p - f}{2 * t} \quad (9)$$

H ranges from 0 to 1. This could be used to compare the performance of a case-base before and after its maintenance.

Based on a framework for categorizing CBM policies (Leake and Wilson, 1998), we describe the approach that is proposed to be semi-automated. The system gathers information about individual cases and the case-base. It is a *diachronic* collection since the policy considers changes in case-base usage over time. Data collection is performed at *periodic* timing with respect to the CBR cycle. The integration is *on-line*. Maintenance is triggered in response to the current size of the case-base. Hence, it is *conditional* and *space-based*. The scope of maintenance is *narrow* since the operation affects a small subset of the case-base.

Conclusions

A relationship between the cases and their entities (feature-values) is established. The similarity function is indicative of the *importance* of a feature towards arriving at a solution and the *significance* of a solution, given a causal value. The *significance* measure is no doubt an enhancement over *idf*, as it weighs the problem and solution parts of stored cases.

The experimental results proved that Nearest Useful Neighbor method improves the retrieval accuracy by considering the past usage of cases. Case usage information is also the criteria for case-base maintenance. It has been described in terms of an established framework.

References

Baeza-Yates R.A. and Ribeiro-Neto B.A., 1999, *Modern Information Retrieval*, ACM Press / Addison-Wesley.

Balakrishnan E., 2000, *Homoeopathy: The Scientific Medicine,* Part I and II, Unicorn Books Private Limited, India.

Bradshaw S., Scheinkman A., and Hammond K., 2000, *Guiding people to information: providing an interface to a digital library using reference as a basis for indexing*, Proceedings of the 5th international conference on Intelligent user interfaces, pp.37-43, New Orleans, Louisiana, United States.

Budzik, J., Hammond, K., and Birnbaum, L., 2001, *Information access in context*, *Knowledge Based Systems,* pp. 37-53.

Duda R. O. and Hart P. E., 1973, *Pattern Classification and Scene Analysis*, Wiley, New York.

Kolodner, J. L., 1993, *Case-Based Reasoning*, Morgan Kaufmann, California.

Lapalme G. and Kosseim L., 2003, *Mercure: Towards an Automatic E-mail Follow-up System*, IEEE Computational Intelligence Bulletin, Vol.2, No.1.

Leake D. B., ed., 1996, Case-Based Reasoning: Experiences, Lessons, & Future Directions, Menlo Park, California, AAAI Press / MIT Press.

Leake D. B., Maguitman A., and Reichherzer T., 2003, *Topic Extraction and Extension to Support Concept Mapping*, In proceedings of FLAIRS-2003, AAAI press.

Leake D. B. and Wilson D. C., 1998, *Categorizing Case-Base Maintenance: Dimensions and Directions*, Advances in CBR, proceedings of 4th European Workshop on CBR, LNCS 1488, pp. 196-207, Springer Verlag.

Lenz M., 1993, *CABATA - A hybrid CBR system*, Pre-prints of the First European Workshop on Case-based Reasoning, pp. 204-209, University of Kaiserslautern.

Mock K., 1996, *Hybrid Hill Climbing and Knowledge Based Techniques for Intelligent News Filtering*, In Proceedings of the 13th National Conference on Artificial Intelligence, Portland, Oregon.

Raghavan P., 1997, *Information Retrieval Algorithms: A Survey*, Proceedings of 8th Annual ACM-SIAM Symposium on Discrete Algorithms, Louisiana.

Salton G., 1989, *Automatic Text Processing: The Transformation, Analysis, and Retrieval of Information by Computer*, Addison-Wesley.