Dimensional indexing for targeted case-base retrieval: The SMIRKS system

Philomena Y. Lee and Michael T. Cox

Department of Computer Science and Engineering
Wright State University
Dayton, OH 45435-0001
{plee;mcox}@cs.wright.edu

Abstract

For a case-based application with a sizable case library, a common practice is to retrieve a set of cases and then reduce the set with a domain-specific similarity metric. This research investigates an alternative technique that constructs a number of two-dimensional (or multi-dimensional) indices using the principle of elaboration and specialization. By storing cases with these indices, we reduce the size of the retrieved candidate set and, in many instances, fetch a single case. This paper describes a case-based reasoner called SMIRKS. We investigate the retrieval performance by comparing linear search, 1-dimensional indexing, and 2-dimensional indexing. The improvement in performance with dimensional indexing is found to be significant, especially in terms of the size of the retrieved candidate set. This paper describes the implementation of SMIRKS, presents the results of evaluation, and discusses some ideas on future applications that can utilize this technique.

Introduction

Cased Based Reasoning (CBR) solves problems by re-using and adapting solutions from past experiences. It has been applied to a wide range of applications such as planning, classification, and speech/image understanding (Kolodner, 1993). A major challenge for CBR is the selection of an appropriate index and the implementation of storage/retrieval schemes so that adaptable cases can be retrieved efficiently and accurately. We address this problem by describing an efficient indexing and retrieval technique that can reduce the size of the retrieved set.

A typical approach to solving the storage/retrieval problem in CBR is to select a set of salient features of the cases as indices (Schank, 1982; Waltz, *et al*., 1989). Unlike the classical database definition, the indices in CBR are defined as one or more discriminating features that act as a cue to retrieve cases. Essentially, an index groups similar cases into a collection thereby facilitating efficient retrieval. For example, consider a script (Schank & Abelson, 1977) that describes a scenario in which a police officer arrests a criminal for illegal drug possession. There are several differentiating characteristics such as: the criminal is a teenage male; the illegal drug is marijuana; and the police perform an arrest. These features can be used as indices to store the case. Later, when a new scenario that describes the drug arrest occurs, we can use the features as probes to locate the previous cases.

Indices are usually established when the case is initially stored, although sometimes indices are dynamically established at retrieval time. Effective index selection can affect whether the system can later successfully retrieve a case. If an index is too general, it will retrieve irrelevant cases. If the indices are too restrictive, the most relevant cases may never be retrieved (Cox, 1994). As more cases are stored, numerous cases may be grouped and retrieved using the same index. A common practice is to select the most appropriate case by a similarity function such as the nearest-neighbor metric based on probability (Short & Fukunago, 1981) and a combined Euclidean-overlap metric (Wilson & Martinez, 1997). This paper describes an alternative technique to narrow the set of candidate cases. This technique is implemented in the Smart Indexed Retrieval for Knowledge Systems (SMIRKS) case-based reasoner.

SMIRKS is implemented by extending the indexing scheme found in Meta-AQUA (Cox & Ram, 1999). Meta-AQUA is a case-based interpreter that can explain and learn from anomalous and interesting events with the goal to understand natural language stories. In general, Meta-AQUA formulates a relationship between a case and its ACTOR attribute. This relationship is used as a top-level index to group together cases that belong to the same category. Other secondary relations are also formed with features that are more specific to differentiate specific cases from the norm (Kolodner, 1984). The indexing system eventually forms a tree with the more general indices close to the root and specialized indices near the leaf nodes. As more cases are acquired, numerous cases are accumulated at the leaf nodes.

To pinpoint the relevant case in the leaf-node, SMIRKS applies the case type (an implicit characteristic that is present in every case) as a separate feature that is orthogonal to the indexing tree. That is, every case is defined as a subtype of another case.[1] These type/sub-type relationships form a hierarchical tree that has a common generic type (i.e., entity) as the root type and is orthogonal to the set of features described before. By superimposing this tree with the index tree on the original feature sets, we have added another dimension to the indexing structures that can be used to narrow the set of candidate cases. Any feature that exists in every case and forms a hierarchical

[1] Such relationships can be considered analogous to the class and subclass of an object in the Object Oriented (OO) methodology.

relationship among all cases, can be used as the second dimension.

A simple analogy can clarify the above description. For example, in the one-dimensional indexing scheme, we have stored a number of cases that describe the police-arrest scenes using the CONTROL feature as index. Such cases include drug possession, burglary, mugging and assault. To retrieve them using the CONTROL as starting relation with "POLICE" and "CRIMINAL-AGENT" as the primary and secondary indices, we will retrieve 4 similar cases as the result. If we implement the two-dimensional scheme in SMIRKS, the 4 cases will initially be classified and stored into subtypes "DRUG-CASE", "BURGLARY", "MUGGING" and "ASSAULT" that are all descendants of a common case type of "CRIMINAL-CASE". On storage, cases of similar types are also grouped together and then indexed using the CONTROL feature. On retrieval, instead of getting all the criminal cases (as in one-dimensional scheme), we only get the cases that are of the same type as the target case. That is, with the new scheme, a search for a drug possession case results in the retrieval of one case instead of all four cases.

In the SMIRKS implementation, we select the case type as such a property because all cases are related to one another with respect to a hierarchy of types. Another reason for our choice is that case elaboration often requires specialization (Kolodner, 1984). We can use this property as a guide for elaboration, generalization and association among related types at various levels to find other relationships.

Generalization and specialization have been applied to many classical CBR systems such as CHEF (Hammond, 1989) and CYRUS (Kolodner, 1984). CHEF creates primary indices on features that represent a set of abstraction descriptors. Then a secondary set of descriptors is used to determine the applicable range of the first set. CYRUS creates a discrimination network of the event types. It then elaborates on the event type to differentiate among the cases. It also applies associations to relate one case with the others. However, there is a subtle difference between SMIRKS and the traditional systems. In SMIRKS, the second dimension (descriptor) used to narrow the search is present in all the cases but it is separate and independent of the features in the primary dimension. This is different from CHEF and CYRUS where the descriptors used to narrow the selection are derived or associated with the primary descriptor. SMIRKS is more analogous to a context-guided retrieval as applied to Hierarchical Case Representation (Watson & Perera, 1997). Watson and Perera restrict case retrieval to children of cases found in earlier steps. In SMIRKS, retrieval using the second dimensional indexes is used to narrow the candidate cases. However, since the higher dimensional indexes are separate and independent, retrieval can also be conducted in other branches that are not the direct descendant of the candidate set.

Conceptually, SMIRKS defines an orthogonal feature (e.g., the case-type) that is inherent and common to all the cases. This feature is merged with the original indexing scheme to form a new index structure.

Implementation

We implemented the above scheme using Meta-AQUA as a framework. Meta-AQUA is a story understanding system that understands and explains anomalies within stories automatically generated by Tale-Spin (Meehan, 1981; see also Cox & Ram, 1999b). It performs the understanding task with various AI algorithms, one of which is a CBR system. The CBR component of Meta-AQUA uses a flat case-library for storage and linear search for retrieval. The one-dimensional indexing scheme described above is used by Meta-AQUA's explanation component to store and retrieve explanations. In order to have a platform to evaluate the performances, we modified the case-based component of Meta-AQUA so that it could use dimensional indexing for retrieval as well as linear search.

The case-based skimming component in Meta-AQUA scans story segments to interpret uninteresting events. It uses a linear match algorithm to find similar scripts from a case base that is initially populated with predefined, hard-coded cases. If there is a match, it skips to the next sentence. If it cannot locate a match, it proceeds to understand the sentence by calling an interpretator (the discussion of which is outside the scope of this paper). SMIRKS replaces the hard-coded predefined cases in the case base with various combination of stories generated by Tale-Spin.

In the one-dimensional scheme, SMIRKS formulates the outcome of the story into a relation that establishes the top index. This in turn generates a second level index with the entity that activates or causes the outcome. Retrieval is performed by using the outcome of the new script to probe the index structure for a similar value that will lead to the relevant cases. Fig. 1 shows the 1-dimensional index structure for a script about an arrest event. The outcome (i.e., arrest) is chosen as the first level index and the entity (i.e., actor) is chosen as second level index that eventually leads to a set of cases.

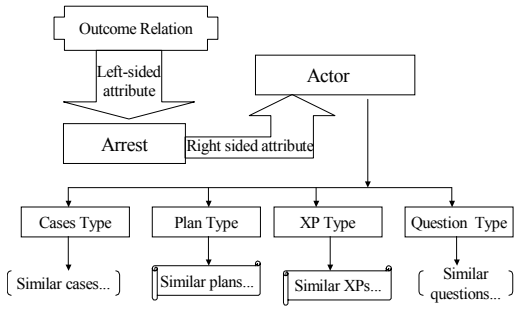

Figure 1
One-dimensional Index

Selection of the alternate indices is important in the 2-dimensional scheme. A suitable feature to be used as the

second dimensional index is one that is present in all the cases. There should also be a hierarchical relationship among the cases based on this feature. In SMIRKS, we take advantage of the "is-a" (or type) property that is possessed by every story. This feature can classify all the stories into a hierarchical tree. All stories follow the type hierarchy while using the "outcome" facet as index. For a story that does not fit into any type hierarchy, it is default to the generic 'case-type.0' that is the parent of all types.

After the knowledge base is initialized, SMIRKS generates a test story and uses it to retrieve similar cases from the knowledge base. The 'main-result' of the story is used as the cue to retrieve matching scripts from the knowledge base. If the knowledge base is established using the 1-dimensional scheme, the cases are retrieved using the same scheme; if the knowledge base is stored using 2-dimensional scheme, then the type of the new script is derived as well as the "outcome" relation. Both the type and "outcome" are then used for retrieval. Fig. 2 shows the 2-dimensional index structure for the same script. It shows a more elaborated type hierarchy after the relationship has been formulated.

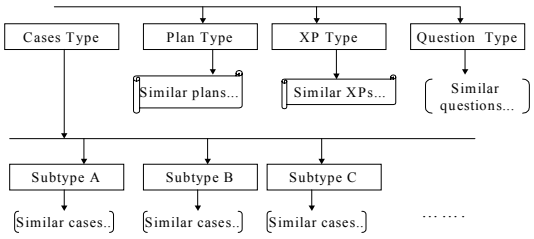

Figure 2.
Two-dimensional Index

Evaluation

SMIRKS compares the performance of linear match, 1-dimensional, and 2-dimensional indexing methods on a Sun Ultr a-60 2-processor (360Mhz) machine with 2 GB DRAM and a 4MB L2 hardware cache running Solaris 8. It populates 7 case libraries with sizes varying from 12 stories to 98 stories.[2] Then it generates a set of test stories and retrieves similar ones using each of the three algorithms. The retrieval is performed on every case library in order to measure data for case bases of different sizes. To reduce the impact of other processes in the machine, SMIRKS measures the performance and computes the mean, standard deviation and mode over the size of the set (i.e., 12). We elected to use the means in the report. We collected the following four types of performance data.

- The average CPU time used in user mode.
- The average wall clock time
- The average number of cons cells being used.
- The average size of the retrieved result-set.

The contents of the case libraries are randomly generated. The identical set of test stories is used for the three algorithms. We define four types of stories with their various subtypes. Table 1 lists the number of stories in each subtype in different case libraries.

Table 1. Case Distribution in the Case Library

Story Types or Subtypes		Case Library Size						
		12	36	48	60	72	84	96
Jonesing	.illegal	1	5	8	10	10	13	16
	.legal	2	6	8	10	12	14	16
Thirsty	.25	1	3	4	5	6	7	8
	.42	1	3	4	5	6	7	8
	.29	1	3	4	5	6	7	8
	.0	1	3	4	5	6	7	8
Drug case	.arrest	1	3	4	2	6	6	8
	.noarrest							
Spirited		4	10	12	15	20	22	24
others					3		1	

Retrieval Time

Fig. 3 compares the User-Mode CPU Usage (excluding garbage collection) that is spent in retrieving cases from the case libraries of different sizes.

Figure 3. Retrieval Time as a Function of Case Size

The data show that the CPU time used in the indexed retrieval is several times less than a linear match is. The 2-dimensional index uses slightly less CPU time than the 1-dimensional index specifically with our biggest case library.

Fig. 4 compares the wall-clock time for the 3 different schemes using the same set of case libraries. The plot shows similar trends as the CPU time. At larger case base, the elapsed time in the 2-dimensional scheme is half that of the 1-dimensional scheme.

[2] The libraries are independent. That is, a case library of size 36 does not contain the smaller library of size 12.

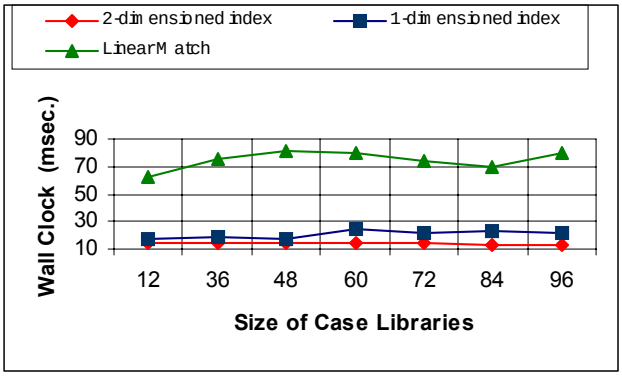

Figure 4. Wall Clock Time as a Function of Case Size

Memory Usage

To compare the memory usage, we measured the number of cons cells used. Fig. 5 shows a plot of the cons cells used by the 3 schemes as a function of the case library size. We found the number of cons cells used by indexed technique is less than that of linear match. The difference between 1-dimensional and 2-dimensional techniques is negligible. It is expected that as the case base grows, more cons cells will be used by the 2-dimensional than by the 1-dimensional indexed technique.

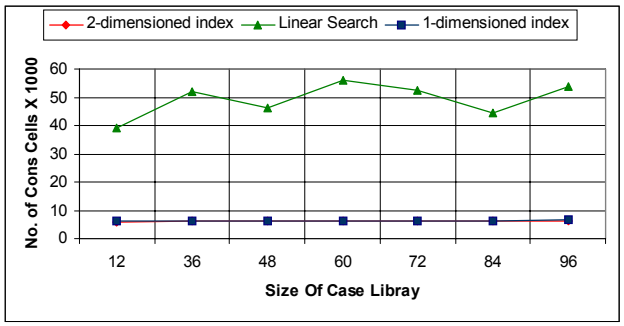

Figure 5. Memory Usage as a Function of Case Size

To accentuate the differences in memory usage between the 2 schemes, we measured the cons cell usage for 1-dimensional indices and 2-dimensional indices *only* in a series of separate runs independent of the linear search. The measurement is plotted in Fig. 6. It indicates that the 2-dimensional index consistently uses more cons cells that 1-dimensional usage.

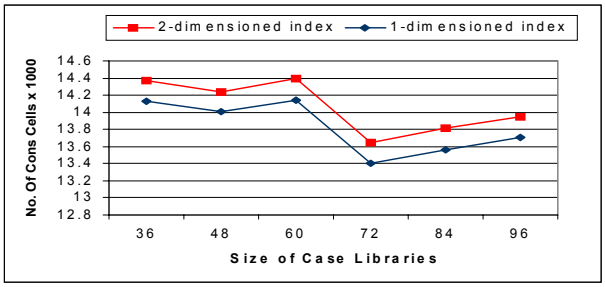

Figure 6. Memory Usage Comparison Without Linear Match

Result Set Size

Another important aspect being considered is the extent to which the new scheme is capable to narrow the size of the retrieved set. So we measured the average size of the retrieved set of similar cases resulting from 1-dimensional and 2-dimenstioned schemes. Fig. 7 is a plot of the average retrieved set size for different case libraries.

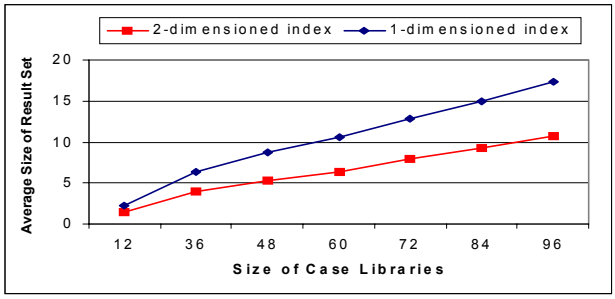

Figure 7. Result Set Size as a Function of Case Size

The data shows that decrease in resultant set size is not significant only for very small case libraries (i.e., at size 12). However, as case base increases, the 2-dimensional indexing scheme retrieves a smaller set than that of the 1-dimensional scheme. These figures are statistically significant at $P = 0.05$ for both ANOVA and Scheffe tests of statistical inference. It is concluded that the 2-dimensional indexing scheme is able to pinpoint similar cases more effectively for large case bases.

Analysis Summary

As expected, the indexed schemes use less CPU time than linear match does. The difference in CPU usage between the 1 and 2-dimensional schemes is not significant. The response time improves for 2-dimensional index as the size of case library increases. This implies that better response time may be expected when we have a larger and more diverse case library. The indexed schemes use less memory than linear match does. The 2-dimensional indices consistently use more memory than 1-dimension. This might imply that the 2-dimensional scheme is scaleable. However with the limited size and uniformity of sample cases, more study is needed to determine the scalability.

The 2-dimensional indexed scheme is quite effective in limiting the size of the result set. The decrease in the number of cases retrieved is proportional to the size of case library. As the size of result set is domain specific, the raw number is not a perfect measure of retrieval precision. However, the relative differences in set size between the two schemes and the fact that these differences increase with size of the case base indicates that the new scheme is quite effective in pinpointing the relevant cases. Moreover, in some applications (e.g., recommender systems), it is desirable to retrieve a diverse set of cases rather than a narrow set. In such system, it is possible to fine-tune the

indices in the second dimension so as to balance the degree of diversity and similarity.

Conclusion

SMIRKS may be enhanced to make the indexing scheme practical for applications. The following list describes a few of the possible near-term enhancements.

- Allow users to select a subtype for classifying the input script before a case retrieval.
- On failure to locate a matched script, change the algorithm to search on subtypes at the same level (siblings) or search up the hierarchical tree.
- Implement multiple inheritances, thereby allowing a case to be classified under more than one sub-type. This will enable the implementation of n-dimensional indices.
- Allow ways for users to customize the number of levels (sub-types) that can exist in the type hierarchy.

In summary, SMIRKS demonstrates that a 2-dimensional index can improve performance significantly in the domain of story understanding. Whether such performance is sustainable in other domains is unknown. More study in a variety of applications is necessary to draw a general conclusion on the benefit of such a method.

Moreover, in certain applications (e.g., recommender systems), it is advantageous to have a larger candidate set with several degrees of diversity (Smyth & McClaire, 2001). Without diversity, such systems may return duplicate choices that are less useful to the user than multiple, varying choices. During adaptation, it is also important to have diversity in the retrieved cases such that different combinations of the features can be considered to generate a creative solution. Although SMIRKS handles methods to pinpoint similar cases, this foundation can be extended to incorporate diversity into a multi-dimensional indexing scheme.

Acknowledgments

The second author is supported by the Air Force Office of Scientific Research (AFOSR) under grant number F49620-99-1-0244 and by a grant from the Information Technology Research Institute (ITRI) at Wright State University**.** The authors would also like to acknowledge the anonymous reviewers.

References

Cox, M. T. 1994. Machines that forget: Learning from retrieval failure of mis-indexed explanations. In *Proc. of the 16th Annual Conference of the Cognitive Science Society*, 225-230. Hillsdale, NJ: LEA.

Cox, M. T., & Ram, A. 1999a. Introspective Multistrategy Learning: On the construction of learning strategies. *Artificial Intelligence*, *112*, 1-55.

Cox, M. T., & Ram, A. 1999b. On the intersection of story understanding and learning. In A. Ram & K. Moorman (Eds.), *Understanding language understanding: Computational models of reading*, 397-434. Cambridge, MA: MIT Press.

Hammond, K. J. 1989. *Case-based planning: Viewing planning as a memory task*. San Diego: Academic Press.

Kolodner, J. L. 1993. *Case-based reasoning*. San Mateo, CA: Morgan Kaufmann Publishers.

Kolodner, J. L. 1984. *Retrieval and organizational strategies in conceptual memory: A computer model*. Hillsdale, NJ: Lawrence Erlbaum Associates

Meehan, J. 1981. Talespin. In R. C. Schank & C. Riesbeck Eds.), *Inside computer understanding: Five Programs plus miniatures* 197-258. Hillsdale, NJ: Lawrence Erlbaum Associates.

Schank, R. C. 1982. *Dynamic memory*. Cambridge, MA: Cambridge University Press.

Schank, R. C., & Abelson, R. 1977. *Scripts, plans, goals and understanding*. Hillsdale, NJ: LEA.

Short, R. D., & Fukunago., K. 1981. The optimal distance measure for nearest neighbor classification. *IEEE Transactions on Information Theory, 27*: 622-627.

Smythe, B., & McClaire, P. 2001. Similarity versus diversity. In D. W. Aha, I. Watson, & Q. Yang (Eds.), *Case-Based Reasoning Research and Development: Proceedings of the 4th. International Conference on Case-Based Reasoning* 347-361. Berlin: Springer.

Waltz, D., Martin, C., Pazzani, M., & Thagard, P. 1989. Panel on indexing algorithms. In *Proceedings of a Workshop on Case-Based Reasoning* 45-65. San Mateo, CA: Morgan Kaufmann.

Wilson, D., & Martinez, T. 1997. Improved heterogeneous distance functions. *Journal of Artificial Intelligence Research, 11*: 1-34.

Watson, I., & Perera, S. 1997. The Evaluation of a Hierarchical Case Representation Using Context Guided Retrieval. In D. Leake, E. Plaza (Eds.), *Case-Based Reasoning Research and Development: Proceedings of the 2nd. International Conference on Case-Based Reasoning* 255-266. Berlin: Springer.