Answering comparison questions in SHAKEN: A progress report

Shawn Nicholson

Qualitative Reasoning Group
Northwestern University
1890 Maple Avenue
Evanston, IL, 60201, USA
nicholson@northwestern.edu

Kenneth D. Forbus

Qualitative Reasoning Group
Northwestern University
1890 Maple Avenue
Evanston, IL, 60201, USA
forbus@northwestern.edu

Abstract

An important class of questions for knowledge based systems concern comparisons, such as "How is *X* like *Y*?" and "How are *X* and *Y* different?" This paper describes how we have used a cognitive simulation of analogical processing to answer such questions, to support domain experts in entering new knowledge. We outline techniques for case construction and summarization of comparison results that have been developed and refined based on an independent formative evaluation. In addition to these techniques, we discuss the role of the comparison system in SHAKEN, the larger system in which they are embedded, and our plans for further improvements.

Introduction

One important task of knowledge-based systems is answering questions. Users ask questions of knowledge-based systems for many reasons: The question itself may be relevant to some larger task that only the user is privy to, the question might relate to some task that the user and software are jointly undertaking, or the question might be asked as a way of seeing what the system understands. The latter is especially important in *knowledge capture*, where the user is a domain expert (rather than an AI expert) who is interacting with the system in order to build up its knowledge base by interacting with it, much as a teacher might interact with a student. In knowledge capture tasks, *comparison questions* are particularly important. Two examples of comparison questions are

"How are *X* and Y similar?"

"How are *X* and *Y* different?"

where *X* and *Y* are two concepts that the system knows something about. The importance of comparison questions arises from several properties. First, comparison provides a context for describing knowledge. Asking how *X* is similar to *Y* focuses on what they have in common, rather than just asking for an exhaustive listing of what is known about *X*. Second, comparison provides a means of highlighting differences. A common trap in knowledge-based systems is reading more into the system's knowledge based on the names given to concepts as opposed to what is actually stated about the concepts. For instance, a system that had concepts for DNAMolecule and RNAMolecule should know that a DNAMolecule has two strands while an RNAMolecule has only one. Third, comparison provides suggestions for additional knowledge that could be entered. Missing differences, as in the previous example, is one source of suggestions. Missing similarities are another: If the system suggests that RNAMolecule might have strands, based on a comparison with DNAMolecule, the user can consider whether or not that suggestion, or some variation of it, is valid.

This paper describes new techniques we have developed for answering comparison questions. These techniques rely on ideas from *structure-mapping* [6], a psychological theory of analogy and similarity. Our algorithms use the *structure-mapping engine* (SME) [1,5,4] as the comparison mechanism. That stable foundation only provides part of the answer, however. Our techniques address three key issues in answering comparison questions:

1. *Case construction.* How should the concept descriptions be automatically generated from the underlying knowledge base?
2. *Evaluating candidate inferences.* How should the inferences generated from the comparison be tested?
3. *Summarization.* How should the results of the comparison be used to generate a helpful answer?

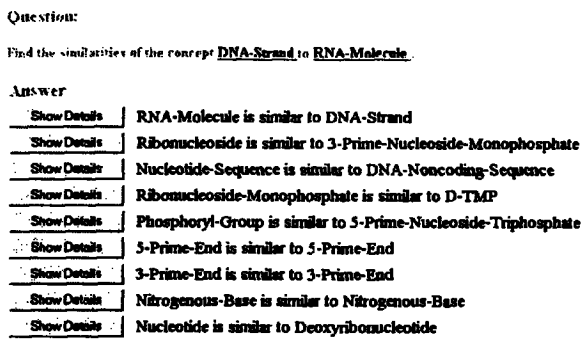
This paper describes how we address these issues in algorithms for answering comparison questions. These algorithms have been integrated into two knowledge formulation systems, and tested successfully (although there is ample room for improvement, as we describe below) with domain experts in independent evaluations. We start by briefly reviewing the relevant psychology and properties of SME. Then we summarize the SHAKEN knowledge formulation system, how our work fits in it, and the formative feedback we received through the evaluation process. Next we describe our algorithms for case construction, evaluating candidate inferences, and summarization, including how we have addressed the problems found through the evaluation process. Finally, we discuss the broader implications of this work.

Structure-mapping: A brief review

Gentner's structure-mapping theory [6] provides an account of analogy and similarity based on comparisons of structured representations. According to structure-mapping theory, structural alignment takes as input two structured representations (*base* and *target*) and produces as output a set of mappings. Each mapping consists of a set of *correspondences* that align items in the base with items in the target and a set of *candidate inferences*, which are surmises about the target made on the basis of the base representation plus the correspondences. The constraints on the correspondences include *structural consistency*, i.e., that each item in the base maps to at most one item in the target and vice-versa (the *1:1 constraint*) and that if a correspondence between two statements is included in a mapping, then so must correspondences between its arguments (the *parallel connectivity* constraint). Which mapping is chosen is governed by the *systematicity* constraint: Preference is given to mappings that match systems of relations in the base and target. Each of these theoretical constraints is motivated by the role analogy plays in cognitive processing. The 1:1 and parallel connectivity constraints ensure that the candidate inferences of a mapping are well-defined. The systematicity constraint reflects a (tacit) preference for inferential power in analogical arguments.

Although structure-mapping started as a model of analogy, it has been extended to model similarity and comparison more generally [7,8]. Some of these results bear directly on the problem at hand. First, the same structural alignment process used in analogy is also used in human similarity judgments, including within-domain comparisons. Thus SME [1,5,4] our cognitive simulation of structural alignment, will provide results that human users should find natural. Second, when asked to describe differences, people use the same structural alignment process to figure out commonalities. This induces a distinction between two types of differences. *Alignable* differences are differences related to commonalities. An example of an alignable difference might be the color of two cars or their horsepower – differences in properties of corresponding arguments. All other differences are *non-alignable* differences. This distinction is important because alignable differences are more psychologically salient than non-alignable differences. Algorithms for answering questions about differences need to exploit this distinction to provide results that their users will find natural.

These psychological results provide important constraints on the process of answering comparison questions, since we need to produce results that will be natural for the human questioners. Since structural alignment is used for comparisons, we can use SME as a component to compare the descriptions of two concepts. The correspondences it produces provide the set of similarities that a person would compute, given the same descriptions. These same correspondences are used in computing alignable differences. Non-alignable differences can be extracted by analyzing the non-overlapping parts of the descriptions mapped. These computations are described in more detail below. Of course, the psychological constraints are only part of the puzzle. The demands of the task and the system that our algorithms are embedded also provide key constraints. We discuss these next.


Question:

Find the similarities of the concept DNA-Strand to RNA-Molecule.

Answer

Show Details	RNA-Molecule is similar to DNA-Strand
Show Details	Ribonucleoside is similar to 3-Prime-Nucleoside-Monophosphate
Show Details	Nucleotide-Sequence is similar to DNA-Noncoding-Sequence
Show Details	Ribonucleoside-Monophosphate is similar to D-TMP
Show Details	Phosphoryl-Group is similar to 5-Prime-Nucleoside-Triphosphate
Show Details	5-Prime-End is similar to 5-Prime-End
Show Details	3-Prime-End is similar to 3-Prime-End
Show Details	Nitrogenous-Base is similar to Nitrogenous-Base
Show Details	Nucleotide is similar to Deoxyribonucleotide

Figure 1: An early example of a comparison answer

Context: Knowledge Formation Systems

One of the biggest bottlenecks to the widespread deployment of knowledge-based systems is the difficulty of building large knowledge bases. DARPA's Rapid Knowledge Formation program is tackling this problem by creating systems that can be used by domain experts to create their own knowledge bases, without the constant intervention of AI experts. This requires the software to communicate with the domain expert in understandable terms: To present what it knows, to assimilate new knowledge, and to enable the expert to test its knowledge by questioning it. Two integrated systems have been developed by teams, one led by SRI and one led by Cycorp. We are supplying analogical processing services for both systems. This paper focuses on our work for the SRI system, called SHAKEN.

SHAKEN uses several methods to make its knowledge understandable to human experts. At the level of content, SHAKEN's knowledge is organized into a library of components that can be composed to create domain-specific descriptions. The components are implemented in a frame system called KM [1]. In terms of display, experts interact with the system via a formalized version of *concept maps*, a technique that has been widely used in education. Browsing and search tools provide navigation aids, and a forms-based interface provides the means of asking questions [2]. Our software is used whenever one of the two comparison questions listed at the start of the paper is asked. The format of our answers evolved during development; Figure 1 shows an example.

The basic algorithm for describing the similarities between two concepts X and Y is

1. Construct case descriptions c(X), c(Y) for X and Y
2. Use SME to compare c(X) and c(Y)
3. Summarize the correspondences found via SME.

The algorithm for describing the differences is very similar:

1. Construct case descriptions c(X), c(Y) for X and Y
2. Use SME to compare c(X) and c(Y)
3. Derive alignable and nonalignable differences based on the SME results
4. Summarize the correspondences and differences

With the exception of step #2 in both algorithms, our algorithms for each step have evolved significantly in response to user feedback. The next section describes our solutions and their evolution, in terms of the three issues introduced in the beginning of the paper.

Our approach

Our techniques can be divided into three categories. First, case construction techniques extract a relevant subset of knowledge about the concepts to be compared from the knowledge base. Second, candidate inference filtering eliminates "obviously wrong" conjectures. Third, summarization techniques simplify the results of the comparison and arrange them in a form that should be easier for users to apprehend. We discuss each in turn.

Case construction

Since the task is knowledge entry, we are comparing general conceptual knowledge from the KB rather than concrete examples. This conceptual knowledge is instantiated for comparison by creating a skolem individual and instantiating knowledge about it. Our initial algorithm was very simple:

Input: a concept X
Output: Set of facts F

1. Retrieve all facts from KM about X
2. For each fact which mentions another entity Y
   a. Add Y to F
   b. If $|F| < N$ (where N is a fixed maximum number of expressions) If Y has not been seen before, recurse, retrieving all facts about Y

The main advantage (and, as it turns out, drawback) of this algorithm is that it did not require more specific contextual knowledge (as used in [9]). Since almost everything in the KB is interconnected, tight bounds had to be drawn. We kept N at about 300 initially, thinking that this would lead to reasonable amounts of information for matching. We discovered that this led to superfluous output, and obscured details that users considered important. The summarization techniques discussed below ameliorated this problem to some degree, but clearly a better case construction method was called for.

How can the system detect what the user thinks is important? Since SHAKEN is designed for knowledge capture, users are entering and browsing knowledge using a concept map display, dynamically expanding and contracting the level of detail shown about aspects of a concept based on what they are doing. We decided to take advantage of the fact that these displays tacitly express what users are currently thinking of as important, and use this information to guide case construction:

Input: a concept map representing concept X
Output: Set of facts F

1. Retrieve the facts and entities from the concept map for concept X as currently displayed.
2. For each entity that is in a leaf position in the concept map, extend it by retrieving all facts from KM that mention that entity.

In other words, the case information extracted is what the user has chosen to display, which presumably is related to what they are thinking about currently, plus one level of expansion at the leaves, to provide more accurate matches for them. We believe that this algorithm will help eliminate the superfluous detail problem. At the very least it provides the user more control over the process, letting them play a more active role.

Evaluating Candidate inferences

Initially we did not perform any evaluation of candidate inferences, due to time constraints. This was a mistake in retrospect, since having suggestions appear that were "obviously" wrong in terms of what the system had already been told eroded trust in the rest of the answer. By contrast, suppose the user knows that the system will only make suggestions that might, as far as it knows, be true. Such suggestions that the user thinks are false provide evidence as to what additional knowledge should be entered, in order to rule them out. This argument suggests that the most important evaluation to be done of candidate inferences is to filter out those which are already known to be incorrect.

Now we use KM-based reasoning techniques to check every candidate inference, and eliminate those which are provably false given the current KB contents. Would more sophisticated evaluation of candidate inferences add value in this task? We believe that the answer is no. During knowledge entry concept descriptions tend to be incomplete, so it seems unlikely that much would be gained by additional validation effort.

Summarizing results

As noted above, answering both similarity and difference questions relies on the correspondences found by SME. In the case of similarity questions, the focus is on the correspondences. For difference questions, the correspondences are used to figure out what differences should be most salient to users, so that those can be presented before other differences.

Similarities are presented in terms of a list of the entities that correspond in the two concepts. SME chooses entity correspondences based on relational overlap (see [5]), so

the correspondences between expressions that suggest an entity correspondence are available via drill-down as a form of explanation. Such explanations by their nature are evidential, often leading to "bushy" justifications[1]. Their size depends on the relative overlap in the relational structure, and provides a useful explanation as to why the match came out the way it did.

One serious problem we found is that, with the large descriptions produced by our original case construction technique, the number of entity correspondences was huge, and overwhelmed users. One useful trick was to cluster entity correspondences by types of entities involved (e.g., those entity correspondences involving strands), which ameliorated but did not resolve the problem. We think that entity-type summarization combined with our new case construction method will make a substantial improvement.

In summarizing differences, since alignable differences are more salient we present them first. An important type of alignable difference are *property differences*, where two entities that play similar roles are of different types. These are detected by examining the attributes that hold for corresponding entities. Like correspondences, drill-down is available to inspect why the matcher found a specific difference interesting. Candidate inferences, computed in both directions, are also summarized as interesting differences. Suppose two concepts are very similar in the user's mind. In that context, a candidate inference is generally either a statement about the other description that needs to be made (in some form or another), or indicates that other knowledge should be added to rule out that possibility.

Non-alignable differences are mentioned last. Non-alignable differences are statements that are true in one description but not the other, with no correspondences in common. In this knowledge capture task, such differences have a similar import to candidate inferences in terms of what they should suggest for the expert.

Evaluation

The first evaluation of SHAKEN occurred in the summer of 2001, as part of the DARPA Rapid Knowledge Formation program. SHAKEN was delivered to IET, an independent contractor, who then had biology graduate students use it to build knowledge bases about textbook biology knowledge. Overall, the results were encouraging. Unfortunately, as indicated above, the original versions of our algorithms did not fare so well.

The evaluators found three major problems:

1. Too much information was presented. This made it difficult for users to understand the analogy and find information that they were looking for. They wanted an even higher level summary of the analogy.
2. Differences that were obviously wrong were included in the explanation.
3. Similarities that users expected were sometimes not found.

We believe that the change in case construction will be a major step towards solving the first problem, since the original algorithm had no way of knowing what users thought was important. However, we suspect that judicious dynamic rerepresentation of the conceptual descriptions, based on the outcome of an initial match, will be needed to completely resolve this problem. The second problem has been addressed by using KM inference to filter candidate inferences, as described above.

The third problem is the most difficult, because it is in part a function of the KM style of representation. Recall that each expression match is a piece of evidence about what entities should be placed into correspondence. There are two factors that work in opposite directions to make matching more difficult. First, the amount of discrimination a relationship provides depends on the size of the relational vocabulary available. If most of the relationships are the same (e.g., `Part-of` in many KM concepts), then there is little reason to choose one match over another, and the choice is driven by attribute information about the entities involved. Second, if experts are less uniform in their representation choices (e.g., using `Part-of` in one description and `Basic-Structural-Unit` for the similar relation in another), the matcher's job is harder. The second problem is easier to solve than the first, in that techniques such as minimal ascension have been used to allow close relational matches. Some of this is simply a function of the concepts not being sufficiently articulated, and can serve as a signal that entering more knowledge to emphasize (or rule out) the similarity between the concepts would be useful.

Although we are not pleased with the summer results, they still constitute valuable information: We learned that the simplest techniques are insufficient, and we learned how our summaries needed to evolve to be more useful to users. All of the changes described in this paper have been implemented in the new version of the SHAKEN system, and will be tested in the midterm evaluation in January 2002[2]. This paper will be updated with those results once they are available.

[1] The difference between such justifications and the more typical low branching factor justifications found in standard dependency structures can be a source of confusion, but since this is the nature of analogical reasoning, the problem is one of training.

[2] From a purely experimental design standpoint, it would be informative to do a sensitivity analysis to see how much improvement each of these changes provides. Unfortunately, such an experiment is far too expensive to carry out in these circumstances.

Discussion

The three issues we addressed here (case construction, filtering candidate inferences, and summarization) are relevant beyond just answering comparison questions: these are issues that any theory of an analogical reasoning task must address. We believe that the solutions we presented here are more broadly relevant. For example, our use of what is effectively discourse context to constrain case construction is very likely to be applicable to any interactive analogical reasoning context, although the evidence as to what is relevant may be harder to extract in less visual interfaces. Similarly, in evaluating candidate inferences, the technique of filtering out conjectures that can cheaply be disproved seems generally applicable. Also, our correspondence-based method of summarizing similarities and the decomposition of differences are reasonable first-cut solutions. However, many issues remain to be explored, especially when other analogical processing tasks are considered. Some of these issues are:

- In case construction, the key questions seem to be (a) how should cues as to the scope of material to include be ascertained given the task context, (b) how much should case construction be driven by incremental outputs of the summarization process, (c) what notions of salience are relevant for a given task?
- In evaluating candidate inferences, the key questions seem to be (a) how much work should be spent disproving a conjecture, (b) should work be spent proving a conjecture, and (c) how should skolems in candidate inferences be resolved?
- In summarization, the key questions seem to be (a) what set of rerepresentation techniques would provide concise, informative human-like summaries and (b) how can drill-down be supported without overwhelming the user?

Answers to these questions will be an important next step in the development of the theory of analogical reasoning.

Acknowledgements

This research is being carried out as part of the DARPA Rapid Knowledge Formation Program. The initial case construction algorithm was based on code written by Peter Clark. We thank Vinay Chaudhri for insightful comments, and the entire SHAKEN team for their excellent work, without which this research would not be possible.

References

1. K. Barker, B. Porter, and P. Clark. A Library of Generic Concepts for Composing Knowledge Bases. *First International Conference on Knowledge Capture*, October 21-23, 2001
2. P. Clark, J. Thompson, K. Barker, B. Porter, V. Chaudhri, A. Rodriguez, J. Thomere, S. Mishra, Y. Gil, P. Hayes, T. Reichherzer. Knowledge Entry as the Graphical Assembly of Components. *First International Conference on Knowledge Capture*, October 21-23, 2001.
3. Falkenhainer, B., Forbus, K., Gentner, D. (1989) The Structure-Mapping Engine: Algorithm and examples. *Artificial Intelligence*, 41, pp 1-63.
4. Forbus, K. 2000. Exploring analogy in the large. In Gentner, D., Holyoak, K. and Kokinov, B. (Eds) *Analogy: Perspectives from Cognitive Science*. Cambridge, MA: MIT Press.
5. Forbus, K., Ferguson, R. and Gentner, D. (1994) Incremental structure-mapping. *Proceedings of the Cognitive Science Society*, August.
6. Gentner, D. (1983) Structure Mapping: a theoretical framework for analogy. *Cognitive Science*, 7: 155-170
7. Gentner, D. and Markman, A.B. (1995) Similarity is like analogy: Structural alignment in comparison. In C. Cacciari (Ed.), Similarity in language, thought, and perception (pp. 111-147). Brussels: BREPOLS.
8. Gentner, D. and Markman, A. 1997. Structure Mapping in Analogy and Similarity. *American Psychologist*, January, pp 45-56
9. Mostek, T., Forbus, K, Meverden, C. (2000) Dynamic case creation and expansion for analogical reasoning. *Proceedings of AAAI-2000*. Austin, TX.