Interpreting Loosely Encoded Questions*

James Fan and **Bruce Porter**
Department of Computer Sciences
University of Texas at Austin
Austin, TX 78712 USA
{jfan, porter}@cs.utexas.edu

Abstract

Knowledge-based question-answering systems have become quite competent and robust at answering a wide range of questions in different domains, however in order to ask questions correctly, one needs to have intimate knowledge of the structure of the knowledge base, and typical users lack this knowledge. We address this problem by developing a system that uses the content of the knowledge base to automatically align a user's encoding of a query to the structure of the knowledge base. Our preliminary evaluation shows the system detects and corrects most misalignments, and users are able to pose most questions quickly.

Introduction

Knowledge-based question-answering systems have become quite competent and robust. Unlike their expert-system predecessors, modern systems answer a wide range of types of questions. Moreover, they cover broad domains (e.g. infectious disease), not just isolated tasks (e.g. diagnosis of infectious disease). For example, several question-answering systems built for Project Halo (Vulcan Inc. 2003) performed well on a targeted portion of the Advanced Placement (AP) exam in chemistry, answering about 160 widely varying and novel questions (Barker *et al.* 2004; Angele *et al.* 2003).

Despite this success, Project Halo revealed a significant problem that limits the utility of question-answering systems: *posing questions to knowledge-based question-answering systems requires intimate knowledge of the structure of the knowledge base, and ordinary users lack this knowledge.* Consequently, their question encodings are structurally flawed, and no state-of-the-art question-answering system will answer the questions correctly.

One solution to this problem, but a false one, is to restrict the expressiveness of the query language to such a degree that only "legal queries" can be posed. For example, conventional information-retrieval systems use a query language that permits only boolean combinations of keywords. Queries posed in this language are matched with target texts, which are little more than bags of words and their frequencies, so there is no opportunity for structural misalignments. However, the keyword-query language is inappropriate for asking conventional questions, such as this typical one from Project Halo:

> When dilute nitric acid was added to a solution of one of the following chemicals, a gas was evolved. This gas turned a drop of limewater, $Ca(OH)_2$ cloudy, due to the formation of a white precipitate. The chemical was:

(a) household ammonia, NH_3
(b) baking soda, $NaHCO_3$
(c) table salt, $NaCl$
(d) epsom salt, $MgSO_4 \cdot 7H_2O$
(e) bleach, 5% $NaOCl$

Another approach to restricting the expressiveness of the query language is to use question templates. The template language permits expressions such as: "what happens to $\langle entity \rangle$ during $\langle process \rangle$?", where $\langle entity \rangle$ and $\langle process \rangle$ are typed variables that can be bound to individuals in the knowledge base. Although question templates have been used successfully (Clark *et al.* 2003; Buchanan & Shortliffe 1984), they have serious limitations. First, the set of templates is small and fixed, which limits their usefulness for novel questions. Second, they lack the expressiveness to describe the context in which the question is posed. For example, the question above arises in the context of a chemical mixture which produces a gas, which in turn mixes with limewater to produce a precipitate.

The goal of our research is to address the problem of posing questions to knowledge-based systems by automating the task of aligning question encodings with the structure of the knowledge base.

Loose speak

We assume that questions originate in English, or some other natural language, and that some person or program encodes them in the formal language used by the question-answering system. If the question is encoded without regard for the structure of the knowledge base, we call it a *naïve encoding*. If the encoding happens to align with the structure of the

*Full support for this research was provided by Vulcan Inc. as part of Project Halo

	Pilot (N = 43)	Final (N = 107)
No loose speak	2.3%	0.0%
Metonymy	25.0%	10.8%
Causal factor	45.5%	23.5%
Role	9.1%	7.8%
Aggregate	77.3%	87.3%
Too generic	34.1%	40.2%
Noun compounds	22.7%	29.4%

Table 1: Types of loose speak, and their frequencies, encountered in two sets of AP-like questions from Project Halo. The pilot set contains 43 questions in total; the final set contains 107 questions in total. The first row shows the percentage of questions that contained no loose speak – i.e. for which a naïve encoding was correct. Note that the sum of the frequencies exceeds 100% because each question may contain multiple occurrences of loose speak.

knowledge base, then it is a *correct encoding* of the question. If, on the other hand, any part of the encoding fails to align with the knowledge base, we call that part *loose speak*, in reference to the imprecise way that people form English expressions.

The term "loose speak" is not meant to be pejorative. To the contrary, question encodings that contain loose speak are often literal translations from the original English expressions– i.e. the form of questions we should expect from typical users. The goal of our project is to improve knowledge-based question-answering systems by developing ways to automate the interpretation of loose speak to produce correct encodings of questions.

Types of loose speak and their frequencies

Misalignments between queries and the knowledge bases to which they are posed are common and unavoidable because knowledge bases, like many engineered systems, are full of idiosyncratic design decisions. Since a naïve encoding may differ from a correct encoding in many ways, developing and testing a loose-speak interpreter appears to be a daunting task.

However, we have found that occurrences of loose speak cluster into a small set of categories that share interpretation strategies. This section describes a study in which we analyzed two sets of questions from Project Halo. All the questions were given in English. We built naïve encodings for the questions and matched the encodings with the structure of the chemistry knowledge base built for Project Halo (Barker *et al.* 2004). Here are the types of loose speak we encountered:

- Metonymy: referring to an entity or event (the referent) by one of its features or attributes (the metonym). The metonym is related to the referent through a restricted set of relations; a commonly used list of metonymic relations is compiled by (Lakoff & Johnson 1980). It has the following relations: PART-FOR-WHOLE, PRODUCER-FOR-PRODUCT, OBJECT-FOR-USER, CONTROLLER-FOR-CONTROLLED, INSTITUTION-FOR-PEOPLE-RESPONSIBLE, PLACE-FOR-INSTITUTION and PLACE-FOR-EVENT.

  Figure 1 contains an example of metonymy: a property ($conductivity$), which pertains to a part (solutes), is ascribed to the whole (the aqueous solution).
- Causal factor: referring to a result by one of its causes. For example, a chemistry question might refer implicitly to a chemical reaction by mentioning only the "mix" or "add" event that causes the reaction.
- Aggregate: referring to an aggregate by one of its members, or *vice-versa*. For example, in figure 1, HCl must refer to a set of molecules, i.e. a chemical, not an individual HCl molecule, because a solution of HCl should contain more than one HCl molecule. Although this type of loose speak is similar to PART-FOR-WHOLE metonymy, it differs in terms of the partonomy types to which it applies. The PART-FOR-WHOLE metonymic relation applies to heterogeneous sets (e.g. an entity can have different types of parts, such as the various parts of a car), but the aggregate relation applies to homogeneous sets (e.g. each individual in the set is a basic unit of the set, such as a single soldier in an army).
- Role: referring to an entity by a role played by that entity, or *vice-versa*(Fan *et al.* 2001). For example, a Bronsted acid is a role played by chemicals in the context of reactions, but is often used loosely, omitting the context as in: "these chemicals are Bronsted acids".
- Overly generic concept: referring to a specific concept or instance by the generic class to which it belongs. For example, a "reaction" is used to refer to an "equilibrium reaction".
- Noun compounds: referring to the implicit relation be-

$$\begin{tikzcd}\text{Aqueous-Solution} \arrow[r, "\text{conductivity}"] \arrow[d, "\text{base}"'] & ? \\ \text{HCl} & \end{tikzcd}$$

Figure 1: An example of loose speak from the Project Halo question set: "Is a water solution of HCl conductive?". The "?" in the figure indicates an query. With respect to the chemistry knowledge base, this encoding contains two occurrences of loose speak. First, it contains aggregate loose speak: a HCl is a molecule, and a single molecule does not act as the solute of a solution because it does not have properties such as concentration. The HCl in the encoding should be replaced by a $Chemical$ whose basic structural unit is a HCl molecule. Second, it contains metonymy loose speak: the $conductivity$ property belongs to solutes instead of solutions because it is the solutes that determine the conductivity of a solution. The query "the conductivity of Aqueous-Solution" should be placed on the base of the Aqueous-Solution instead. Figure 4 shows the correct encoding of this query.

```
Given a triple of the form <C1, r, C2>:

  RESULT = nil
  If domain_violation (C1, r) then
    RESULT = search_head(<C1,r,C2>)
  else if range_violation (C2, r) then
      RESULT = search_tail(<C1,r,C2>)
    else if not(resemblace_test (<C1, r, C2>)) then
        RESULT = search_head(<C1,r,C2>)
        If not (RESULT) then
          RESULT = search_tail(<C1,r,C2>)
        end if
      end if
    end if
  end if
  If RESULT then
    Return RESULT
  else
    Return <C1, r, C2>
  end if
```

Figure 2: The search algorithm used to interpret one triple. The algorithm detects loose speak by checking the domain, range constraints and checking to see if the input expression is similar to any existing knowledge. If loose speak is found, then it calls the two search routine to search for encodings similar to the input, and it returns the search result if any is found, or the original input if nothing else is found.

tween a pair of nouns by using just the noun phrase consisting of the two nouns. For example, "zinc metal" refers to metal *composed of* zinc. Noun compounds are a type of loose speak because they omit the semantic relation between the head noun and its modifier(s), and this omission must be rectified before the reference is clear.

Table 1 reports the results of our study. We can draw several conclusions from this study. First, loose speak is ubiquitous, occurring in virtually every question we analyzed. Second, loose speak concentrates on a small number of types. We found common types of loose speak, some of which occur as frequently as 87% of the time.

Interpreting loose speak

Algorithm Overview

The input to our loose-speak interpreter is a question encoding, which may or may not contain loose speak, and the output is the correct encoding of the question. The question encoding is a conceptual graph consisting of a set of labeled edges of the form: $\langle ConceptClass, relation, FillerClass\rangle$. We call $ConceptClass$ the ***head*** and $FillerClass$ the ***tail***. Queries are encoded as a special type of edge in which the tail is unknown (denoted: "?"). Figure 1 is an example of such encoding.

The interpreter makes multiple passes through the set of edges. Each pass is equivalent to traversing the graph. During each pass, the interpreter applies a "test & repair" function to every edge. The interpreter stops when a pass is completed in which all the edges pass the test, so none is repaired. The order of traversal does not matter because (other than "overly generic concept" type of loose speak) the interpreter does not alter the original head or tail. Rather, the interpretation of each edge is a series of connecting edges, and the interpretation of one edge does not affect the interpretation of other edges. For "overly generic concept" type of loose speak, the interpreter replaces the head or the tail of an edge with a more specific class, and a simple backtracking algorithm is used in case the more specific class conflicts with edges connected to the original edge.

Test & repair

During the traversal of the conceptual graph encoding the question, each edge is tested and repaired. Figure 2 shows the algorithm for the test and repair procedure. The test determines whether the edge contains loose speak. Because the correct encoding is unknown, the test uses this heuristic:

> If the edge violates structural constraints, then it must contain loose speak (because correct encodings are consistent with the structure of the knowledge base).

The only structural constraints used by the interpreter are based on the domain and range of the relation in an edge. Specifically, for an edge $\langle C_1, r, C_2\rangle$, the domain of r must subsume C_1, and the range of r must subsume C_2.

Although the constraint check returns many true positive cases, it also may return many false negatives. The constraint check is complemented by a resemblance check. The resemblance check is based on the hypothesis that if the input does not resemble any existing knowledge, then it may contain loose speak because studies have shown that "one frequently repeats similar versions of general theories" (Clark, Thompson, & Porter 2000). Even though the knowledge resemblance test may return many false positives, it returns many true negatives as well.

The resemblance test, applied to an edge, is implemented as follows. If the edge represents a query (i.e. the tail of the edge is "?"), the edge passes the test only if the knowledge base can compute one or more fillers for the tail. (This is the basic question-answering function of a knowledge base.) Otherwise, the edge $\langle Head_q, r, Tail_q\rangle$ passes the test only if the knowledge base contains an edge $\langle Head_{kb}, r, Tail_{kb}\rangle$, such that $Head_{kb}$ subsumes or is subsumed by $Head_q$ and $Tail_{kb}$ subsumes or is subsumed by $Tail_q$.

After loose speak has been detected in a edge, it is interpreted by searching the knowledge base for similar edges. The search is conducted by two breadth-first procedures: `search_head` and `search_tail`. Given the edge $\langle C_1, r, C_2\rangle$, the `search_head` function starts at C_1, traverses all semantic relations, such as "has-part", "agent" and "subclasses", and stops when a suitable instance is found. An instance, C_3, is suitable if the triple $\langle C_3, r, C_2\rangle$ does not contain loose speak. The successful search path is returned as the interpretation for the input.

To avoid unintended interpretations, the semantic relations do not include "superclasses" relation. If both "subclasses" and "superclasses" are included in the search path, then any concept can be found from C_1 by climbing up and

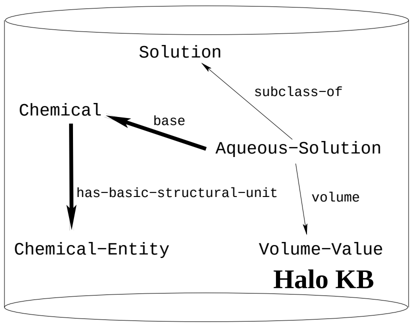

Figure 3: The application of "search-head" function on the edge $\langle$ *Aqueous-Solution*, *base*, HCl $\rangle$. The function is called because this edge does not resemble any existing path in the knowledge base. The loose speak is resolved by breadth-first search from *Aqueous-Solution*, along all the semantic relations such as *volume*, *base*, *subclasses*, until a HCl or a superclass or subclass of HCl, is found. In this example, the search stops at depth 2, and returns the new edge $\langle$ *Aqueous-Solution*, *base*, $\langle$ *Chemical*, *has-basic-structural-unit*, HCl $\rangle$ $\rangle$ because HCl is a superclass of *Chemical-Entity*. The bold line in the above figure shows how the result is found.

down the taxonomy, and a large number of spurious interpretations may be returned for the given input.

If multiple suitable instances are found, then they are ordered based on their distances from C_1. Either users are prompted to choose one correct interpretation from the list of results or the first result is automatically chosen.

If the original input has a domain constraint violation and no result is found, then an additional search is performed. The additional breadth-first search starts from an instance of the domain class, and it searches along all semantic relations until C_1 is found.

The `search_tail` function is implemented similarly as a search that starts at C_2.

The search is conducted in a breadth-first manner for two reasons. First, because each occurrence of loose speak is a naïve encoding that has a limited number of differences from its correct encoding, a correct interpretation should be closely related to the naïve input. A very deep search will only return encodings that are not closely related to the input. If only a shallow search is needed, then a brute force search such as breadth-first search is sufficient. The maximum search depth in our system is set 4. Second, a breadth-first search returns the shallowest interpretation without examining the whole search space. Such breadth-first searches have been shown to be effective in interpreting noun compound type of loose speak given a top level ontology that contains the fundamental axioms (Fan, Barker, & Porter 2003).

If no interpretation is found when the depth cutoff is reached, the interpreter returns the original encoding. In addition, if loose speak was detected by a constraint violation, the interpreter returns an error report.

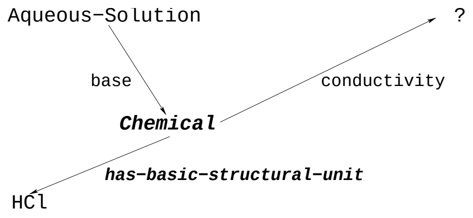

Figure 4: The interpretation of the query in Figure 1 in the context of the chemistry knowledge base built for Project Halo. Notice that HCl is replaced by *Chemical*, and the *conductivity* query has been moved from *Aqueous-Solution* to the *base* of *Aqueous-Solution* to conform to the structure of the knowledge base.

This algorithm applies to all types of loose speak, even noun compounds. A noun compound is represented as an unlabeled edge of the form: $\langle ModifierClass, nil, HeadClass\rangle$. Because there is no unlabeled edges in the knowledge base, noun compounds will fail the resemblance test, and consequently trigger `search_head` and `search_tail`.

Example

For the example in figure 1, the loose-speak interpreter traverses the query graph to test & repair each edge. The edge $\langle$ *Aqueous-Solution*, *base*, HCl $\rangle$ is tested first. The domain of *base* is constrained to be a *Solution*, and the range is constrained to be a *Thing*. Both constraints are satisfied because *Aqueous-Solution* is a subclass of *Solution* and HCl is a subclass of *Thing*. However, there is no path in the knowledge base that resembles the edge, so the `search-head` function is called to search from the head, *Aqueous-Solution*. The search returns *Chemical*, whose *has-basic-structural-unit* is *Chemical-Entity* (the superclass of HCl). See Figure 3. Consequently, the edge is interpreted as $\langle$ *Aqueous-Solution*, *base*,*Chemical* $\rangle$.

The edge $\langle$ *Aqueous-Solution*, *conductivity*, ? $\rangle$ is then tested for loose speak. No constraints are violated, but the query does not yield any result. This prompts a breadth-first search from *Aqueous-Solution*. This search stops at *Chemical*, the *base* of *Aqueous-Solution*, because it has a conductivity value and is a suitable replacement for *Aqueous-Solution* in the original edge.

Figure 4 shows the final interpretation of the query graph. Notice that the solute of the *Aqueous-Solution* is replaced by *Chemical*, and the *conductivity* query has been moved from *Aqueous-Solution* to *Chemical*. After these changes, the two types of loose speak, metonymy and aggregate, are resolved in the query graph and the encoding is correctly aligned with the structure of the knowledge base.

Experiment

To evaluate our approach, we asked several users to encode a set of questions, and compared their naïve encodings with the corresponding output from the loose-speak interpreter.

	KE Experience	Chemistry Background	Experiment length	Questions encoded	Correct encodings	Interpreter precision	Interpreter recall
User A	0.5 yr	medium	10.5 hr	47 (94%)	3 (6.4%)	97.4%	90.5%
User B	17 yrs	weak	9.95 hr	48 (96%)	4 (8.3%)	100%	90.7%
User C	0.25 yr	medium	10 hr	44 (88%)	5 (11.4%)	96.8%	87.8%

Table 2: The evaluation results of the loose-speak interpreter. The first two columns show users' backgrounds in knowledge engineering and chemistry. The next three columns show users' performance: the time required to finish the experiment; the number of questions encoded; and the number of correct encodings. The last two columns measure the interpreter's performance in term of precision recall. As the data have shown, despite different backgrounds, all users were able to encode most of questions, the encodings had many occurrences of loose speak, the interpreter was able to correct most encodings, and to do so with high precision.

Setup

For this experiment we used a set of fifty multiple choice questions from various mock AP chemistry tests. This set was compiled by a chemist (anonymous to us), who also provided the answers for the questions. This data set is distinct from the two sets of questions used in the frequency study, and it is distinct from the training data that we used to refine the interpreter.

We recruited three users, with varying backgrounds in knowledge engineering and chemistry, to pose questions to the chemistry knowledge base built for Project Halo. All three were familiar with KM (Porter & Clark 1998), the knowledge representation language used in the experiment, but none was familiar with the chemistry knowledge base and its structure. Their knowledge engineering experience varied from a few months to many years. Their chemistry backgrounds varied from weak (high school chemistry decades ago) to medium (college chemistry years ago).

The users were given a short (three page) tutorial on encoding questions. It contains two sections: format and vocabulary. The format section explained that question encodings should have a context part, which gives the contextual information of a question, and an output part, which specifies the value being sought. The format section illustrates this with three examples. The vocabulary section lists commonly used classes and slots in the chemistry knowledge base, and their corresponding chemistry terms. Users were also pointed to a web page containing the complete listing of slots and classes, and they were allowed to take as much time as necessary to complete the questions. These brief instructions were not a complete tutorial on using the knowledge base, as evidenced by the high rate of loose speak in the users' encodings of questions.

We used metrics of precision and recall to evaluate the interpreter's performance. Precision and recall are defined as follows (Jurafsky & Martin 2000):

$$Precision = \frac{\#\ of\ correct\ answers\ given\ by\ system}{\#\ of\ answers\ given\ by\ system}$$

$$Recall = \frac{\#\ of\ correct\ answers\ given\ by\ system}{total\ \#\ of\ possible\ correct\ answers}$$

For our experiment, the correct answers are the number of question encodings that are interpreted correctly. Answers given by the system are the number of question encodings for which the interpreter detects loose speak and finds an interpretation. Finally, all possible correct answers are the number of all the question encodings that contain loose speak. An interpretation is determined to be correct if it completely aligns with the chemistry knowledge base and conveys the intended semantics of the question encoding as judged by us. Precision estimates the likelihood of a correct interpretation when an interpretation is found; recall is a measurement of coverage.

Our experiment does not include noun compound type of loose speak for two reasons. First, the effectiveness of the interpreter on noun compounds has been evaluated separately in (Fan, Barker, & Porter 2003). Second, although noun compounds, such as "carbon monoxide", occur frequently in chemistry questions, their interpretations do not require inferring a variety of semantic relations, therefore chemistry is an unrepresentative domain for evaluating noun compound interpretation.

For this experiment, we ran the interpreter in "batch mode". If the interpreter found multiple interpretations for any question, we treated the first interpretation as its sole response.

Data analysis and discussion

The experimental results are shown in table 2. The first two columns give the users' background in knowledge engineering and in chemistry; the next three columns record the time it took users to encode the test questions, the number of questions they were able to encode, and the percentages of correct encodings. The last two columns measure how well the interpreter performed.

Based on the data, we can draw three conclusions:

1. Loose speak is very common – just as we found in our preliminary study (described in the "Types of loose speak and their frequencies" section). Only 4% of the encodings on average contain no loose speak. This underscores the importance of the loose-speak interpreter for building useful knowledge-based question-answering systems.

2. The loose-speak interpreter works well. As shown in Table 2, the precision of the interpreter is above 95%, and the recall is around 90%. This shows the search algorithm we used is suitable for interpreting naïve encodings.

In addition we found that 54% of the loose speak occurrences are detected by constraint violations, and the rest by the resemblance check. This suggests both checks are important for detecting loose speak.

3. The loose-speak interpreter significantly assists novice users in posing questions. The results show that there is no clear correlation between users' knowledge engineering experience and the number of questions they are able to encode. Users with little knowledge engineering experience were able to encode large percentages of questions quickly. This is good news; it suggests that with the burden of loose-speak interpretation lifted, knowledge engineering is no longer a significant hurdle to question-answering. Users can concentrate on the content of questions instead of knowledge engineering details.

Related work

Metonymy is one type of loose speak that has been studied extensively in the computational linguistic community. There are two basic approaches to metonymy interpretation. First, there are systems that interpret metonymic expressions based on a set of rules (Weischedel & Sondheimer 1983; Grosz *et al.* 1987; Lytinen, Burridge, & Kirtner 1992; Fass 1997). These rules contain explicit knowledge about how to interpret some types of metonymy, and they are matched against inputs to produce interpretations. This approach is easy to implement because no large knowledge base is required, but it can handle only a fixed set of metonymy types. Second, there are systems that interpret metonymic expressions based on knowledge-base search (Browse 1978; Markert & Hahn 1997; Harabagiu 1998). Unlike the rule-based systems, they do not contain explicit knowledge of different types of metonymy, instead they rely on searches in a general purpose knowledge base. Our loose speak interpreter is one of them. Although this approach has no limit on the types of metonymy it can handle, the demand for an extensive knowledge base makes it difficult to implement. Because our interpreter is used for knowledge-based question-answering systems, it avoids this cost by operating on an existing knowledge-base.

Another closely related linguistic phenomenon is the use of generics, a technique called "genericity". The study of genericity deals with how to distinguish between classes and instances (Carlson & Pelletier 1995). For example, "potato" refers to the class of potato and an instance of potato in the following two sentences.

a. The potato was first cultivated in South America.

b. The potato was rotten.

This is similar to the aggregate type of loose speak because an individual is used to refer to an aggregate, the class. However aggregate type of loose speak differs from genericity when the aggregate being referred to is not the whole class of concepts. For example, in figure 1, HCl refers to a particular set of molecules that is dissolved in the solution, not the complete class of HCl molecules. Most of the genericity detection methods are either syntax based or require human judgment. They are not suitable for automatic interpretation of loose speak in a knowledge-rich environment such as question-answering.

Techniques similar to knowledge-base search are used for other tasks, such as knowledge acquisition, in addition to loose-speak interpretation (Davis 1979; Kim & Gil 1999; Blythe 2001). Many of these systems build explicit meta-knowledge representation, such as the tree of rule models in (Davis 1979), and use such meta-knowledge to guide knowledge acquisition. Our loose-speak interpreter does not have meta-knowledge representations, instead it uses generic search algorithm directly on the knowledge base itself to find useful knowledge to assist question-answering.

Conclusion

In this paper, we defined loose speak as the part of a question encoding that misaligns with existing knowledge base structures, and we presented a system that would interpret loose speak automatically. Our preliminary evaluation shows that loose speak is common in question encodings, and that our interpreter can detect and interpret most occurrences of loose speak correctly. Without the burden of manually interpreting every occurrence of loose speak, users were able to pose a good percentage of questions quickly.

In the near future, we will expand the investigation of loose speak into other aspects of knowledge base interaction, such as knowledge acquisition, and evaluate its impact.

References

Angele, J.; Moench, E.; Oppermann, H.; Staab, S.; and Wenke, D. 2003. Ontology-based query and answering in chemistry: Ontonova@project halo. In *Proceedings of 2nd International Semantic Web Conference*.

Barker, K.; Chaw, S. Y.; Fan, J.; Porter, B.; Tecuci, D.; Yeh, P.; Chaudhri, V. K.; Israel, D.; Mishra, S.; Romero, P.; and Clark, P. E. 2004. A question-answering system for AP chemistry: Assessing KR&R technologies. In *Principles of Knowledge Representation and Reasoning: Proceedings of the Ninth International Conference*.

Blythe, J. 2001. Integrating expectations from different sources to help end users acquire procedural knowledge. In *Proceedings of Seventeenth International Joint Conference on Artificial Intelligence*.

Browse, R. A. 1978. Knowledge identification and metaphor. In *Proceedings of the 2nd Biennial Conferene of the Canadian Society for Computational Studies of Intelligence (CSCSI-2)*, 48–54.

Buchanan, B. G., and Shortliffe, E. 1984. *Rule-Based Expert Systems: The MYCIN Experiments of the Stanford Heuristic Programming Project*. Reading, MA: Addison-Wesley.

Carlson, G. N., and Pelletier, F. J., eds. 1995. *The Generic Book*. Chicago: University of Chicago Press.

Clark, P.; Barker, K.; Porter, B.; Chaudhri, V.; Mishra, S.; and Thomere, J. 2003. Enabling domain experts to convey questions to a machine: a modified, template-based approach. In *Proceedings of Second International Conference on Knowledge Capture*.

Clark, P.; Thompson, J.; and Porter, B. 2000. Knowledge patterns. In *Principles of Knowledge Representation and Reasoning: Proceedings of the Seventh International Conference*.

Davis, R. 1979. Interactive transfer of expertise: Acquisition of new inference rules. *Artificial Intelligence* 12(2):121 – 157.

Fan, J.; Barker, K.; and Porter, B. 2003. The knowledge required to interpret noun compounds. In *Proceedings of Eighteenth International Joint Conference on Artificial Intelligence*.

Fan, J.; Barker, K.; Porter, B.; and Clark, P. 2001. Representing roles and purpose. In *Proceedings of First International Conference on Knowledge Capture*.

Fass, D. 1997. *Processing Metonymy and Metaphor*. Greenwich, Connecticut: Ablex Publishing.

Grosz, B. J.; Appelt, D. E.; Martin, P. A.; and Pereira, F. C. N. 1987. TEAM: An experiment in the design of transportable naturual-language interfaces. *Artificial Intelligence* 32(2):173–243.

Harabagiu, S. 1998. Deriving metonymic coercions from wordnet. In *Proceedings of the Workshop on Usage of WordNet in Natural Language Processing Systems, COLING-ACL*, 142–148.

Jurafsky, D., and Martin, J. 2000. *Speech and Language Processing: An Introduction to Natural Language Processsing Computational Linguistics, and Speech Recognition*. New Jersey: Prentice Hall.

Kim, J., and Gil, Y. 1999. Deriving expectations to guide knowledge-base creation. In *Proceedings of Sixteenth National Conference on Artificial Intelligence*. AAAI Press.

Lakoff, G., and Johnson, M. 1980. *Metaphors We Live By*. Chicago: University of Chicago Press.

Lytinen, S. L.; Burridge, R. R.; and Kirtner, J. D. 1992. The role of literal meaning in the comprehension of non-literal constructions. *Computational intelligence* 8(3):416–432.

Markert, K., and Hahn, U. 1997. On the interaction of metonymies and anaphora. In *Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence*.

Porter, B., and Clark, P. 1998. KM - the knowledge machine: Reference manual. Technical report, University of Texas at Austin. http://www.cs.utexas.edu/users/mfkb/km.html.

Vulcan Inc. 2003. Project Halo. http://projecthalo.com.

Weischedel, R. M., and Sondheimer, N. J. 1983. Metarules as a basis for processing ill-formed input. *American Journal of Computational Linguistics* 9(3-4):161–177.