A Template Matcher for Robust NL Interpretation

Eric Jackson, Douglas Appelt, John Bear, Robert Moore, and Ann Podlozny

SRI International
333 Ravenswood Avenue
Menlo Park, CA 94025

Abstract

In this paper, we describe the Template Matcher, a system built at SRI to provide robust natural-language interpretation in the Air Travel Information System (ATIS) domain. The system appears to be robust to both speech recognition errors and unanticipated or difficult locutions used by speakers. We explain the motivation for the Template Matcher, describe in general terms how it works in comparison with similar systems, and examine its performance. We discuss some limitations of this approach, and sketch a plan for integrating the Template Matcher with an analytic parser, which we believe will combine the advantages of both.

Introduction

One of the conclusions SRI has drawn from working with the ATIS common task data is that, even with a very constrained user task, there will always be unanticipated expressions and difficult constructions in the spoken language elicted by the task that will cause problems for a conventional, analytical approach to natural-language processing. However, it also seems that requests for only a few types of information account for a very large proportion of the utterances produced by users performing a task like air travel planning. This point is illustrated by some of the more difficult queries in the June 1990 test set:

> Give me a list of all airfares for round-trip tickets from Dallas to Boston flying on American Airlines.
>
> Show me all the flights and their fares from San Francisco to Boston on June second.
>
> I need information on airlines servicing Boston flying from Dallas.

In the first example the phrase "flying on American Airlines" apparently modifies "tickets," with the flights that the tickets are for apparently being the implied subject of "flying." The second example seems to contain a discontinuous constituent, "flights .. from San Francisco to Boston on June second," which is the antecedent of the pronoun "their" that occurs in the middle of the discontinuity. The third example would be straightforward, except for the fact that the verb "servicing" has been substituted for the more conventional "serving." Despite the difficult linguistic problems posed by these queries, the information they request is very simple—just fares, flights, and airlines for travel between a pair of specified cities.

Consideration of examples such as these has led us to modify our approach to natural-language processing in spoken language systems. The key modification to our system is the addition of a Template Matcher to provide robust interpretation for the most common types of requests in the task domain. The Template Matcher achieves robustness in two ways: (1) it provides an interpretation when not all the words or constructions in an utterance have been accounted for, and (2) it provides a mechanism for trading-off the risk of wrong answers with the degree of coverage. These properties arise from a mechanism that assigns scores to interpretations, penalizing interpretations that do not account for words in the utterance. The bulk of this paper is devoted to describing the Template Matcher and discussing its performance as a stand-alone system for interpretation of natural-language queries for the ATIS task. Later in the paper we consider how such a module might best fit into a complete system for spoken-language understanding.

Description of the System

The Template Matcher operates by trying to build "templates" from information it finds in the sentence. Based on an analysis of the types of sentences observed in the ATIS corpus, we devised four templates that account for most of the data: flight, fare, ground transportation, and meanings of codes and headings. We have recently added several new templates, including aircraft, city, airline, and airport. Templates consist of slots which the Template Matcher fills with information contained in the user input. Slots are filled by looking through the sentence for particular kinds of short phrases. For example, "from" followed by an airport or city name will cause the "origin" slot to be filled with the appropriate name. The sentence

Show me all the United flights Boston to Dallas nonstop on the third of November leaving after four in the afternoon.

would generate the following flight template:

```
[flight,[stops,nonstop],
        [airline,UA],
        [origin,BOSTON],
        [destination,DALLAS],
        [departing_after,[1600]],
        [date,[november,3,current_year]]]
```

The template score is basically the percentage of words in the sentence that contribute in some way to the building of that template. Given an input sentence, the Template Matcher constructs one template of each sort, and the one with the best score is used to construct the database query, provided its score is greater than a certain "cut-off" parameter. The cut-off parameter is what permits the risk trade-off mentioned above: the higher the cut-off, the more conservative the system is in attempting to produce a response. Words can contribute to a score in different ways: words that fill a slot (e.g., "Boston") add to the score, words that help get a slot filled (e.g. "from") also add to the score. Some words may not contribute to the interpretation, but nonetheless confirm the choice of a particular template (e.g., "downtown" for the ground transportation template), and hence are added to the score for that template. Other words are ignored for the purposes of scoring (e.g., "and," "please," "ok," and "show"), since they do not tend to confirm particular templates.

In certain cases the Template Matcher may modify the basic score of a template. Each template has a set of key words (or key phrases). The presence of these words or phrases in a sentence is a strong indication that the associated template is the appropriate one for that sentence. For the flight template, the keywords include words like "flight," "fly," and "go"; for the fare template, words and phrases such as "how much," "fare," and "price" are examples; for the meaning template, examples include "what is," "explain," and "define." If none of a template's key words are present in a sentence then that template's score is docked by a certain keyword punishment factor, which varies from template to template. In most cases the lack of a keyword will prevent the associated template from scoring above the cut-off.

There are two situations in which the Template Matcher will "abort" a given template, that is, give it a score of zero and cease processing it. First, if the system tries to fill a slot in a certain template with two different values, that template is aborted. Since we have no better than a fifty-fifty chance of guessing which is the correct filler, we are better off not attempting any answer. Second, if a template has no slots filled, it will receive a score of zero. This restriction is relaxed when the Template Matcher is operating in "context-dependent" mode, where follow-up questions are expected. A query like "show me the fares," which would not fill any slots, would be much more likely as a follow-up question than as a context-independent query.

Comparison with Other Systems

Systems using the basic idea behind the Template Matcher go back as least as far as the SAM system at Yale [2], and include the Phoenix system at CMU [3, 4] and the SCISOR system at General Electric [5] as recent examples. There is also a degree of similarity to "case-frame"-based parsing methods [6, 7]. The main distinction is that the slots in our templates are domain-specific concepts rather than general linguistic or conceptual cases.

Of these precursors, the Phoenix system seems most similar to the Template Matcher. Like the Template Matcher, the Phoenix system has templates (which they call "frames") with slots that get filled with information from the sentence. The scoring mechanisms of the two systems are similar, but not identical. For both, the basic score of an interpretation is the number of words in the sentence that the interpretation accounts for. In the Phoenix system, for a word in a sentence to count for an interpretation's score, it must help fill some slot in that interpretation's frame. For the Template Matcher, the word will also count if it is an "ignore" or "confirm" word as discussed above.

There are several other differences between the scoring mechanisms of the two systems: The Template Matcher punishes templates that do not have a keyword present in the sentence, and the Template Matcher requires that at least one slot in a template be filled. Also, the two systems behave differently when an attempt is made to fill a single slot with two different fillers. The Template Matcher will abort a template if this happens, while the Phoenix system will fill the slot with the second of the two possible fillers. The latter approach will handle certain types of false starts, but might be expected to yield more incorrect answers in other situations. Finally, CMU is not currently using a cutoff to weed out bad interpretations, although given the existence of a scoring mechanism in their system, this is something they clearly could do.

Results

After two weeks of development this system was tested on the June 1990 ATIS test set. This was a fair test to the extent that the implementor of the matching routines and the templates themselves (Jackson) had not examined the data from this test set prior to the evaluation. (Moore had noted, however, that the test set queries seemed amenable to a template-matching approach). For various values of the cut-off parameter we obtained the results shown in the following table.

Cut-off	Right	Wrong	No Answer
0.000	55	13	22
0.833	42	4	44
1.000	37	2	51

(These results were determined by visual inspection of the templates; the database retrieval code was not implemented at this point.) The conclusion we drew from this test is that a template-matching approach could quickly yield results that were competitive with the some of the better results reported in the original June 1990 ATIS test.

After completing the implementation of the system and extensive development using the ATIS training data, we used the Template Matcher for the February 1991 ATIS class A evaluation, in both the NL and SLS tests. The results as measured by NIST are shown below.

Test	Right	Wrong	No Answer
NL only	109	9	27
SLS	96	11	38

We used a cut-off of 0.8 for this evaluation, as we had previously determined from training data that this value should come close to optimizing the number of right answers minus the number of wrong answers.

The system for the SLS tests was a serial connection of the version of SRI's DECIPHER system used in the ATIS SPREC evaluation and the Template Matcher described above. The answers reported in the SPREC evaluation were edited to be in lexical SNOR format and run through the Template Matcher exactly as in the NL tests. It is interesting to note the relatively small degradation from the NL to the SLS results, despite a 18.0 percent word error rate in the speech recognition; this seems to indicate the robustness of the Template Matcher to recognition errors.

We had not planned to participate in the D1 evaluation, but at the request of NIST, we did those tests as well, taking context into account by using the answer to the first query in the D1 pair to restrict the database search in answering the second query, the same technique used in our ATIS demo system. In addition, the Template Matcher was run in context-dependent mode for the second query of each D1 pair. The results on the second queries of the pairs as measured by NIST are shown in the table below.

Test	Right	Wrong	No Answer
NL only	22	3	13
SLS	15	11	12

We have not yet analyzed why there was a greater degradation in going from the NL to the SLS results in the D1 tests.

Limitations

In this section, we discuss some sentences that cause problems for the Template Matcher that are not easily resolvable.

> Show me flights returning from Dallas into San Francisco by ten P M.

This sentence is a good example of the need for syntactic information. The problem is that the Template Matcher cannot tell that the phrase "by ten P M" modifies "returning," and thus constrains the arrival time. By default, it treats the "by" phrase as restricting the departure time, and thus misinterprets the query.

> What is an A fare?

The problem here is that "A" is ambiguous; it may be either the indefinite article or a fare class code. We have been forced to leave the fare class code "A" out of the Template Matcher lexicon. Adding it would do more harm than good, for we would then misinterpret every occurence of the phrase "a fare" (with the indefinite article), as in "Give me a fare from Boston to Dallas." Syntactic information could help resolve this ambiguity, as could speech information, since the determiner "a" and the letter "A" have different acoustic properties.

> List the fares for Delta flight eight oh seven and Delta flight six twenty one from Dallas to Denver.

Conjunctions of complex noun phrases are beyond the scope of the Template Matcher as it currently stands. The system could be modified to handle such phenomena, but an analytical grammar might be the more natural tool for the job.

> Do you have to take a Y N flight only at night?

This is an example of a sentence where all the words contribute to a certain template (the flight template, in this case) and yet that template is not the correct one.

A New Architecture

As the examples in the previous section suggest, the Template Matcher by itself is probably not the ultimate solution to the problem of robust interpretation of natural-language queries. We believe that the template-matching approach and an analytical parser-based approach have complementary strengths and that an approach that combines both of them is likely to be ultimately superior than either one alone. We have therefore begun developing a new architecture for language processing in spoken language systems that combines the two approaches. Our basic strategy will be to use the analysis produced by the parser whenever we can, but to fall back on the Template Matcher when the parser-based system fails to produce a complete analysis. It is our conjecture, supported at least in part by the best results reported in the June 1990 ATIS evaluation, that an analytical, parser-based approach can be designed so that when it succeeds in providing a complete analysis of the input, that analysis has a very high probability

of being correct. With the Template Matcher it seems that there will inevitably be a larger possibility for error, because it uses strictly less of the information available in the utterance than a parser. In particular, our Template Matcher can ignore words; it ignores order; and it has almost no notion of structure. By using the Template Matcher as a backup to the parser-based system, we eliminate the possibility of the Template Matcher getting a wrong interpretation of something that could be successfully analyzed by the parser.

A second reason for running the Template Matcher after the parser is to enable the Template Matcher to use partial results of parsing in its operation. Our current Template Matcher uses only single words and fixed phrases as key words or slot fillers. We are in the process of extending the Template Matcher so that it uses whole phrases that have been identified by the parser in attempting to analyze the entire utterance. For example, we saw that the Template Matcher is unable to analyze a phrase as complex as "returning from Dallas into San Francisco by ten P M." Generalized to work from parsed phrases, the Template Matcher might be able to successfully interpret a complex utterance containing this phrase even if the entire utterance could not be parsed. Additionally, running the Template Matcher on parsed phrases should cut down on the sheer number of particular word patterns that have to be included in the template specifications.

The use of robust interpretation methods changes the way in which the constraints embodied in a grammar are viewed. They must be treated as soft, rather than hard, constraints. This has significant implications for the rest of a spoken language system. If we want the parser to find grammatical fragments of the input that may be of use to the Template Matcher, then the parsing algorithm we previously used, which imposed strong left-context constraints, is no longer appropriate. We want something closer to pure bottom-up parsing to find all the phrases that the Template Matcher might use. We have developed such a parser, whose details are outlined in another paper for this workshop [1].

Perhaps the most significant consequence of using robust interpretation methods in a spoken language system, however, is that the failure to find a complete parse can no longer be used as a hard constraint to reduce perplexity for the speech recognizer. An analytical grammar still contains valuable information that should be used by the recognizer, however. We feel that one promising approach to making use of this information is to extend the idea of a word-based statistical language model, such as a bi-gram model, to a phrase-based statistical language model, e.g., a "bi-phrase" model. The idea is simply to estimate the probability of occurrence of a particular type of phrase conditioned on the type of phrase that precedes it. In making this work effectively, however, it is important to include some lexical information in the categorization of phrases, usually information about the lexical head of the phrase.

The ability of such a framework to capture long distance constraints not captured by N-gram models is illustrated by an utterance such as "What airlines that serve Boston fly 747s?" If we want to predict the likelihood of "fly" occuring in this context, the preceding word "Boston" gives us essentially no information. If, however, we have identified "What airlines that serve Boston" as a noun phrase whose lexical head is "airlines" then the likelihood of a verb whose lexical head is "fly" should be relatively high.

The incorporation of a probabilistic element into the system raises a number of other interesting possibilities, including incorporation of probabilistic scoring based on observations of likelihoods of particular templates for sentences in the corpus, of particular slots for each template, and of particular words for each slot; and the possibility of using the Template Matcher itself as the basis of a statistical language model to guide recognition.

Summary

In sum, the Template Matcher represents a complementary approach to traditional natural-language processing. It has the virtues of robustness and broad coverage of many linguistic variants for requests for specific types of information. Although we have not discussed the issue of computational efficiency in this paper, the Template Matcher is noticably faster than a typical parser. The approach also has the advantage of rapid development time which should enhance portability to new domains.

Acknowledgments

This research was supported by the Defense Advanced Research Projects Agency under Contract N00014-90-C-0085 with the Office of Naval Research.

References

[1] Moore, R.C., and Dowding, J., *Efficient Bottom-Up Parsing*, Proceedings, Fourth DARPA Workshop on Speech and Natural Language, February 1991.

[2] Schank, R.C. and Yale A.I. Project, *SAM—A Story Understander*, Research Report 43, Department of Computer Science, Yale University, 1975.

[3] Ward, W., *Understanding Spontaneous Speech*, Proceedings, DARPA Speech and Natural Language Workshop, February 1989.

[4] Ward, W., *The CMU Air Travel Information Service: Understanding Spontaneous Speech*, Proceedings, DARPA Speech and Natural Language Workshop, June 1990.

[5] Rau, L.F., and Jacobs, P.S., *Integrating Top-Down and Bottom-Up Strategies in a Text Processing System*, Proceedings, Second Conference on Applied Natural Language Processing, Austin, Texas, 1988.

[6] Riesbeck, C., and Schank, R.C., *Comprehension by Computer: Expectation-Based Analysis of Sentences in Context*, Research Report 78, Department of Computer Science, Yale University, 1976.

[7] Carbonell, J.G. and Hayes, P.J., *Recovery Strategies for Parsing Extragrammatical Language*, Technical Report CMU-CS-84-107, Carnegie-Mellon University Computer Science Technical Report, 1984.