USING SEMANTICS TO CORRECT PARSER OUTPUT FOR ATIS UTTERANCES

Sheryl Young

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

ABSTRACT

This paper describes the structure and operation of SOUL, or Semantically-Oriented Understanding of Language. SOUL is a knowledge intensive reasoning system which is opportunistically used to provide a more thorough, fine grained analysis of an input utterance following its processing by a case-frame speech parser. The SOUL postprocessor relies upon extensive semantic and pragmatic knowledge to correct, reject and/or clarify the outputs of the CMU PHOENIX case-frame parser for speech and speech transcripts. Specifically, we describe briefly both some of the linguistic phenomena which SOUL addresses and how SOUL works to correct inaccurate interpretations produced by the PHOENIX parser. Finally, we present the results on four separate, non-overlapping test sets. Our "pilot" test sets include the June 1990 DARPA ATIS0 test set and two test sets composed of unseen ATIS0 data distributed in June 1990 that, unlike the DARPA test sets, contain unrestricted utterances. Our forth test set is the official DARPA February 1991 ATIS1 test set. These evaluations illustrate the decrease in error rate that results from SOUL's semantic and pragmatic postprocessing are most pronounced in unrestricted, as opposed to carefully constrained test sets. Specifically, a performance comparison between unrestricted and restricted test sets in pilot experiments show that error rates are reduced by 84% as opposed to 54% when no utterances are pruned from the speaker transcripts.

OVERVIEW

The DARPA speech and natural language community has recently adopted the domain of air travel information for its spoken language systems. The domain has been named ATIS, for air travel information system. Training and test data are gathered for this common task by employing speakers to verbally interact with a database in order to solve one or more randomly assigned, predefined problems with predefined goals and preferences. To perform their task, speakers must verbally query an air travel database. Speakers are not required to use complete, well formed or syntactically correct utterances. They can ask for any information, regardless of whether the request is reasonable, or whether the information contained in the database. Hence, true spontaneous speech is generated. Data is collected by recording both subject utterances and database responses as speakers perform these verbal tasks. The recorded data is then divided into training and test sets of use by the DARPA community.

Thus far, the utterances selected for evaluation test sets are highly constrained, being restricted to those utterances which can be answered using the information in the database and can either be interpreted and answered in isolation, or by using only the context of a preceeding utterance. All utterances that are ambiguous, refer to objects and actions not included in the database, request information that is not available, or contain spontaneous speech phenomena such as mid-utterance oral edits and corrections are removed from the official test sets.

For these evaluations, we designed the SOUL system to enhance the performance of the CMU ATIS system when operating in isolated or limited context modes. The system operates in an opportunistic manner. It is only called upon to perform post processing when there is reasonable uncertainty in the case-frame output. This uncertainty can result from large regions of unaccounted-for speech, multiple, competing interpretations and seemingly incomplete or un-meaningful interpretations. Input to SOUL is the utterance, all the words and phrases matched by the case-frame parser, PHOENIX, and a set of hypothesized interpretation frames (or instantiated case-frame). SOUL outputs either an error message (e.g. No information in database on BOULDER) or a single interpretation composed of corrections, deletions, and all forms of modifications to the instantiated case-frame It does this by using a large semantic and pragmatic knowledge base in conjunction with abductive reasoning and constraint satisfaction techniques. A by-product of the SOUL design is that it also provides much of the semantic and pragmatic knowledge required for our complete dialog and prediction facilities, previously called MINDS system (1988, 1989a,b, 1990) [1, 2, 3, 4].

As SOUL was designed to deal with all spontaneously generated utterances, we expect there will be some advantages for using the system while processing the required, highly restricted data, but that the system will be far more valuable when processing unrestricted input. To evaluate the effectiveness and relative payoff of SOUL, we investigated the following two issues.

First, we wanted to see how much, if any impact use of a large semantic and pragmatic knowledge base would have in reducing error relative to a semantically based (by definition) case-frame parser [5, 6, 7] when processing only isolated utterances or utterances that can be interpreted using only very limited context. Caseframe parsers employ both semantic and syntactic knowledge. They do not use an extensive knowledge base or inferencing procedures. However, they have proven to be very robust and effective in producing interpretations of both well formed and ill-formed user input.

Secondly, we wanted to determine if the use of a knowledge base alone would allow us to process all types of utterances, including those which are un-answerable, request information not in the database and outside the definition of system capabilities, are ambiguous, as well as to be able to detect those utterances which can only be answered by using unavailable contextual information.

To evaluate these questions, we assessed performance of our case-frame speech parser, PHOENIX, both with and without SOUL on four independent test sets. Two of the test sets contained unrestricted data -- every utterance generated was processed. The other two test sets (DARPA ATIS0 and ATIS1) contained restricted utterances, as described above.

The remaineder of this paper describes how SOUL uses semantic and pragmatic knowledge to correct, reject and/or clarify the outputs of the POENIX case-frame parser in the ATIS domain. The next section summarizes some of the linguistic phenomena which SOUL addresses. The following section briefly summarizes how it works to correct inaccurate parses. The last section presents the results of four performance evaluations which contrast the performance of the PHOENIX case-frame parser with and without the SOUL postprocessor.

LINGUISTIC PHENOMENA EXAMINED

SOUL was developed to cope with errors produced by the CMU PHOENIX speech and transcript parsing software in the ATIS domain. Specifically, SOUL augments the basic, rapid pattern matching and speech recognition functions of the PHOENIX system with knowledge intensive reasoning techniques for more fine grained analysis of the preliminary alternative interpretations. Initially we analyzed the performance of the PHOENIX system on a set of training data. The data consisted of 472 utterances comprising dialogs b0 through bn of the ATIS0 training set. An evaluation of the performance of the original PHOENIX system on this data revealed that PHOENIX experienced difficulties with the following problematic linguistic phenomena, which composed a total of 44.3 percent of the utterances: (Note: underlined information not in database)

Unanswerable queries, no information in database, or illegal action requested) (Found in 19.3% of sentences in the training corpus)

What ground transportation is available from the airport in Denver to Boulder at three pm on the twenty second? How do I make reservations? Show all the flights from Dallas to Fort Worth Interpreting these utterances requires knowledge on the limitations of the database, detection of user misconceptions and constraint violations as well as the ability to recognize and understand information not contained in the database.

Context dependent utterances (9.2%)

Show me all returning flights To process isolated utterances or utterances that are only allowed to be interpreted using limited contextual information, it is helpful to be able to recognize those utterances where critical information cannot be reasonably inferred.

Ungrammatical and ill-formed utterances (3.0%)

What date does flight eight seventy seven from San Francisco to Dallas leave from? Ungrammaticality is a part of spontaneous speech. However, one can also obtain ill-formed or ungrammatical input from misrecognition of an input string. These phenomena preclude using a strict syntactic constraints and clues such as definite reference or any type of case marker such as those typically used in textual case-frame parsers.

Ambiguous queries (6.4%)

What's the distance from San Francisco to Oakland? The example query can be interpreted as meaning the city San Francisco or San Francisco International airport. In the case of the former, no information is contained in the database. In the absence of disambiguating context, it is important to be able to recognize all interpretations.

Yes/No and Quantified Yes/No's (3.2%)

Do all of the flights from Pittsburgh to Boston serve meals? These, as well as the next category of utterances require that the critical information be detected from the input. However, they are not problematic when accurately recognized from the speech input.

Superlatives and Comparatives (3.2%)

What's the cheapest flight from Atlanta to DFW?

SOUL was designed to provide a fine grained analysis of input in an opportunistic manner by relying upon a large knowledge base. It was also tuned to "pay attention" to the above listed linguistic phenomena that posed problems for the original version of the PHOENIX speech processing case-frame parser. Furthermore, it was also designed to address some of the problems inherent in spontaneously uttered input.

THE CMU SYSTEM: OVERVIEW

The CMU ATIS System is composed of three interacting modules; a speech recognizer (SPHINX), a robust case-frame parser adapted for spontaneous speech (PHOENIX), and a semantic and pragmatic processor (SOUL) which can work either on isolated utterances or can incorporate the dialog and prediction functionality of the MINDS system. For results reported in this paper, SPHINX produces a single output string which is then processed by the speech case-frame parser, PHOENIX (Ward, 1990). The PHOENIX case-frame parser builds plausible parses of the input string by using robust slot filling heuristics. All possible case-frame slots associated with any portion of an input utterance with a reasonable recognition probability are filled. Then candidate case-frames are built, bottom up, which try to account for as much of the matched input as possible. Once candidate interpretations are generated, PHOENIX either sends all interpretations to SOUL or else, when operating in the absence of SOUL (or when an unambiguous interpretation exists), selects the interpretation which accounts for the most input. In either case, one interpretation is selected. This interpretation is then put into a cannonical form and mapped into an SQL^{TM} database query. This query is then passed to the database and output is presented on a computer screen to the user.

THE SOUL SYSTEM

SOUL relies on a semantic and pragmatic knowledge base to check for consistency in the output interpretations produced by the parser. There are three special features about this frame-based system. First, SOUL not only defines legal values, attributes, and concepts within the ATIS domain, but it also accounts for much extra-domain information as well. Second, it uses inheritance and reasoning to determine contextual constraints, so consistency and constraints can be maintained for combinations of information never before seen. Third, the system uses a single reference data structure for determining illegal input (for which action is prohibited) and unanswerable input (for which no information is in the database).

REASONING

The mechanisms underlying SOUL's abilities are the use of constraint satisfaction techniques in conjunction with what has been called abductive reasoning [8, 9] or concretion [10]. These are general, domain independent techniques which rely upon a domain specific knowledge base. The abductive reasoning component is used to evaluate alternative or candidate phrase matches and to decide which phrases modify one another and to determine which can be put together to form one or more meaningful utterances.

To illustrate, consider the following utterance: **"Does B stand for business class"**. The case-frame parser instantiates the following sequence of concepts or cases: *B = abbreviation B = code B = letter Does = list Does = Explain business class = class-of-service class = class-of-service stand-for = mean.*

The knowledge base is faced with three basic concepts: B, which is an instance of some abbreviation. Specifically, B is an abbreviation for Breakfast and for Business-Class. B can also be a letter, which can be part of a flight number identifying the airline carrier, or part of the call letters of an aircraft, or one of the letters composing the name of a person reserving a ticket. Stand-for indicates equivalence, a specific predicate that is either true or false. Business-class is an interpretation preferable to class alone, as it is more specific. Given an equivalence predicate and a specific concept business-class, the only allowable interpretation of "B" is that its an abbreviation. Even in the absence of an equivalence predicate, there is no additional information which would support the interpretation of "B" as being part of a flight number, carrier identification, aircraft call number or a person's name. Now, given "Business-class", an instance of a fare class, an equivalence relationship and a choice between alternative abbreviation expansions for B, the only expansion which would make the predicate true is the instance of B abbreviating "Business-class".

CONSTRAINT REPRESENTATION AND USE

The abductive component not only determines what possible phrases compose a meaningful request or statement, it also spots combinations which violate domain constraints. Examples of the types of constraint violations which are recognized include violations on both type constraints of objects and attributes as well as n-tuple constraint violations.

To illustrate, consider the following rule for long range transportation taken from the current knowledge base: *Objects: long-range vehicle, origin-location, destination-location inanimate/animate-objects to-be-transported.* Here we have constraints not only on the type of objects that may fill these roles (and of course information about these objects is contained in other portions of the knowledge base) but we have relational constraints as well. The following are the single constraints on the objects involved in long-range transportation. Vehicles are constrained to be included in the instances of a long range vehicle. These include airplanes, trains and cars that are not taxi or limosines. The origin and destination are constrained to be either airports (or their abbreviations) or locations that must include a city (and may include additional information such as state and/or location within or relative to the city. In this example, there is a single relational, or tuple constraint. It poses restrictions on the relationship between the origin and destination slot fillers. These include: If two cities are involved, they cannot be the same, and there must be a set difference between the listings of the airports that service the two cities. If two airports are involved, they must not be the same airport. If a city and an airport are involved, the city must be served solely by the airport listed. Under these rules, you cannot fly from Dallas to Fort Worth in this database. However, you can fly from San Francisco to San Jose. Similar rules for short-range transportation would rule out taking a taxi from Pittsburgh to Boston.

These types of definitions for events, actions, objects, etc. and the constraints placed upon them also allow one to determine whether or not there is sufficient information upon which to take an action. Hence, if a flight is not clearly delineated given whatever context is allowable under the test set rules, these rules can determine whether a query is context dependent or insufficiently specified.

EXAMPLES

The following examples from the ATIS corpus further illustrate how the reasoning component operates.

What is the shortest flight from Dallas to Fort Worth? PHOENIX would look for flights from Dallas to Fort Worth, assuming a correct interpretation. However, SOUL knows that Dallas and Fort Worth are both served only by DFW airport. Since you cannot takeoff and land in the same place, this is an illegal and unanswerable request.

How much would it cost to take a taxi from Pittsburgh to Boston? PHOENIX recognizes "How much would it cost from Pittsburgh to Boston", and would output the corresponding list of flights and fares. SOUL recognizes that "to take a taxi" is important information that has not been included in the interpretation.

It also knows that taxis are short-range transportation vehicles. If the request were legal, SOUL would tell PHOENIX to add taxi as method of transportation and delete airplanes as the transportation vehicle. However, this request violates the constraint on what constitutes a short-range trip, so SOUL outputs the violation type to the speaker (or generates an error message as the CAS).

Are there any Concord flights from Dallas to Boston? Here PHOENIX find a request for flights between Dallas and Boston. SOUL tells the parser to "add Aircraft-Class aircraft_code SSC".

Show all the flights to San Francisco on August 18. Here SOUL recognizes that a departure location has been omitted and cannot be found in any unaccounted-for input. Hence, this is a context-dependent sentence.

SOUL OUTPUT

SOUL takes as input the candidate parses as well as the input string. The output is a list of instructions and codes which are sent back to PHOENIX so they can be incorporated into the database query. Specifically, SOUL outputs an existing interpretation augmented by the following information:

- When there is missing, critical information, database tables and bound variables are output.
- When there is a reasonable selection of information interpreted under an incorrect top level frame (e.g. air travel vs ground travel) it provides a correct top level interpretation or frame.
- When a specific word string is mis-interpreted, it provides corrections in the form of additional variables and their bindings to add as well as variables and bindings to delete.
- When a query involves information not included in the current, restricted database, when the query requires information that is out of the chosen domain, and when the user is asking the system to perform a function it is not designed to do, it outputs specific error codes to PHOENIX indicating what the problem is and outputs specific corrective information to the user screen. This is designed to correct user mis-conceptions. (e.g. Show all flights from Dallas to Fort Worth).
- Finally, when two un-related queries are merged into a single utterance, the system outputs a "break point" so PHOENIX can re-parse the input into two separate requests. (For example the utterance *Show all flights from Philladelohia to Boston and how far Boston is from Cape Cod* would be divided up where as *Show all flights from Philladelphia to Boston as well as their minimum fares.* would not.

To summarize, SOUL looks for and corrects the following types of problems: (1) Information that is missing from the output of PHOENIX, which is added by SOUL. When too much information is missing, the system produces the "do-not-understand" response. (2) Constraint violations. The speaker is informed what is unanswerable, or the parser is given instructions on how to reinterpret the input. (3) Inaccurate parses, where SOUL tells the parser any combination of a basic interpretation frame, information to add, information to delete, or regions to reparse for a specific meaning or variable. (4) Unanswerable queries and commands, which produce a message to the speaker describing what cannot be done.

EXPERIMENTAL RESULTS

PILOT STUDIES

The current implementation of SOUL was trained on the first two-thirds of the ATIS0 training data available in June 1990, consisting of Dialogs B0 through B9 and BA through BN. The training set contained 472 utterances. SOUL was evaluated on the following three independent, non-overlapping test sets. Set 1 contained the 94 Class A and context-removable utterances from the official June 90 ATIS0 test set. Sets 2 and 3 both used sentences from Dialogs BO through BZ from the June 1990 data. These were set aside for use as an independent test set. What is important about this data, is that unlike Set 1, and the February 1991 official DARPA test set (Set 4, described later), the data are not restricted in any manner. All utterances produced by the speakers are included in the test set, regardless of whether they are well formed, within the bounds of the domain, ambiguous, context dependent, etc. Set 2 included all 232 utterances that were not context dependent, and therefore contained unanswerable, ambiguous, ill-formed and ungrammatical utterances, as well as Class A and context-removable queries. Set 3 consisted of the remaining 29 context-dependent utterances contained in the transcripts from Dialogs BO through BZ.

Results of the three evaluations, comparing the performance of PHOENIX alone and PHOENIX plus SOUL are given in the table below. These results were obtained using the standard DARPA/NIST scoring software. However, we allowed for a variety of additional error messages which were more specific than the generic NIST errors. Results using Test Set 1, indicate SOUL's ability to detect and correct inaccurate and incomplete output from the PHOENIX parser, since these sentences consist only of answerable, legal and non-ambiguous utterances. As these utterances are constrained, it is expected that only minor improvments will result for the addition of SOUL. In contrast, Test Set 2 contains unrestricted input, namely all utterances generated which are interpretable without context. Results using Test Set 2 indicate SOUL's ability to recognize unanswerable, derive multiple interpretations for ambiguous input, to interpret ill-formed and un-grammatical input and to correct inaccurate output from the PHOENIX parser. Finally, results from Test Set 3 indicate SOUL's proficiency in detecting context dependent utterances. However, it should be noted that the Test Set 3 results are not representative of POENIX performance. PHOENIX is designed to process context dependent utterances only when using context.

Results from the Three Test Sets					
Test Set	System	Correct	Incorrect	Error Rate	Improvement
Test Set 1	PHOENIX	75	19	20.20%	N/A
	PHOENIX + SOUL	85	9	9.28%	54.1%
Test Set 2	PHOENIX	154	74	32.46%	N/A
	PHOENIX + SOUL	215	13	5.70%	83.98%
Test Set 3	PHOENIX	0	29	100.00%	N/A
	PHOENIX + SOUL	20	9	30.00%	70.0%

Official February 1991 Results: Inclusive of Interface Bugs					
Test Set	System	% Correct	% Incorrect	% No Answer	Total Error
TYPE A	PHOENIX	80.7	16.6	2.8	19.3
	PHOENIX + SOUL	80.7	11.7	7.6	19.3
D1	POENIX		13	3	60.5 %
	PHOENIX + SOUL	114	8	7	70.9 %

Query Type	Error Type	Number	% of Total
Type A	Real (Semantics / Syntax)	12	8.28
	Interface to Phoenix	3	2.07
	Backend (CAS Field + Dumps)	13	8.97
Type D1	Real (Semantics / Syntax)	2	15.79
	Interface to Phoenix	4	5.26
	Backend (CAS Field + Dumps)	5	13.16

RESULTS: FEBRUARY 1991

For the February 1991 official evaluation, we modified the PHOENIX and SOUL systems somewhat, so that all the routines for processing superlatives, comparatives and yes/no utterances were moved into the PHOENIX system. Therefore, the results presented in this section differ from the Test Set #1 in the pilot study section above in that all increases in accuracy due to properly interpreting superlatives, comparatives and yes/no questions are no longer included in the SOUL results.

The February 1991 test set contained 148 isolated utterances and 38 utterances which are interpreted using limited context, or context provided by the preceeding query and its answer. These 148 individual utterances were constrained to be "Class A", or answerable and unambiguous. On the 38 "D1" or limited context queries, if the interpretation or the database response produced in response to the first utterance is inaccurate, the tested utterance will probably not be correct. The results of the evaluation are presented below.

A number of interfaces were modified after performing the pilot evaluation and just prior to performing the February 1991 evaluation presented above. As a result, we introduced a number of errors into the system. The following table breaks down the sources of error and the percentages attributable to each source.

As seen in Table BQ3 of the 19.3% errors on Class A queries, 9% are due to back-end bugs and an additional 2% are due to bugs in the interface between the PHOENIX and SOUL modules. Hence, 12 out of the 19.3 % errors are due to back-end type interface bugs.

For Class D1 queries, of the 34.21% error, 18.42 are due to interface problems, a number reasonably proportional to the Class A results. Here, 5.26% is found in the PHOENIX SOUL interface and 13.16 in the database interface.

All in all, the real errors made by the system for Class A queries result in roughly an 8% error rate, where for Class D1 queries, the error rate is roughly 16%, exactly double they Class A error rate. This is to be expected, as an error made on the context setting portion of a query will necessarily result in an error on the context dependent portion of a set of queries.

An analysis of the official results including interface bugs and/or of the results excluding interfaces bugs indicates that SOUL is responsible for a sight decrease in overall error rate. The results also indicate that SOUL is reasonably good at detecting and flagging inaccurate interpretations, even when it is not able to produce a correct interpretation.

Official February 1991 Results: Interface Bugs Ignored					
Test Set	System	% Correct	% Incorrect	% No Answer	Total Error
TYPE A	PHOENIX	89.65	7.58	2.76	10.35
	PHOENIX + SOUL	91.72	4.14	4.14	8.28
D1	POENIX	78.95	15.79	5.26	21.05
	PHOENIX + SOUL	84.21	5.26	10.52	15.78

The February 1991 official results modified by only the back-end bugs are presented below. These data are derived from the original and official complete log of the February 5, 1991 data run.

SUMMARY

To summarize, SOUL was designed to deal with ambiguous, unanswerable, illegal and context removable utterances. The approach taken was to create an extensive semantic and pragmatic knowledge base for use in abductive reasoning and constraint satisfaction. The resulting system performs fine grained analysis of an input utterance when criteria for activating the postprocessor is met. It was hypothesized that the SOUL processor would contribute more significantly in difficult processing / interpretation situations, while the case-frame parser, itself semantically based, would be sufficient for more restricted test sets. This is shown by comparing error rates of PHOENIX alone and PHOENIX coupled with SOUL across the two conditions of restricted and unrestricted utterance sets. Test Sets 1 and 4 (DARPA June 1990 and February 1991) are highly constrained, while Test Sets 2 and 3 are completely unconstrained. As hypothesized, the SOUL system contributes significantly more to reducing error rates and enhancing accuracy when applied to the more difficult, unrestricted data. When processing unrestricted input, as required in real world applications, the addition of a semantic and pragmatic postprocessor for performing fine grained analyses results in significant improvements in accuracy.

However, it would be expected that given a complete dialog and all the context knowledge a system can capitalize upon, or even a greater amount of context, SOUL would perform better than it does with limited context D1 or no-context Class A utterances even if they were constrained.

The second question posed was whether a knowledge base alone would enable detection of contextually dependent utterances where the applicable context is unavailable. Results of Test Set 3 indicate reasonable detection abilities (70%).

In summary, semantic and pragmatic knowledge can be effectively used to enhance a system's accuracy of interpretation rates. This effect holds even in isolated utterance processing tasks, which provide a great deal less data than can be derived from a complete dialog. In the absence of dialog, the accuracy improvements are more marked in more difficult processing conditions than when processing constrained, relatively straight forward utterances.

REFERENCES

1. Hauptmann, A. G., Young, S. R. and Ward, W. H., "Using Dialog Level Knowledge Sources to Improve Speech Recognition", *Proceedings of the Seventh National Conference on Artificial Intelligence*, Morgan Kaufmann, 1988.

2. Young, S. R., Hauptmann, A. G. and Ward, W. H., "Layering Predictions: Flexible Use of Dialog Expectation in Speech Recognition", in *IJCAI-89*, Morgan Kaufmann, 1989.

3. Young, S. R., Hauptmann, A. G., Ward, W. H., Smith, E. T. and Werner, P., "High Level Knowledge Sources in Usable Speech Recognition Systems", *Communications of the ACM*, Vol. 32, No. 2, 1989, pp..

4. Young, S. R., "Use of Dialog, Pragmatics and Semantics to Enhance Speech Recognition", *Speech Communication*, Vol. 9, 1990, pp..

5. Carbonell, J. G. and Hayes, P. J., "Dynamic Strategy Selection in Flexible Parsing", *ACL81proc*, ACL81, 1981.

6. Hayes, P. J. and Carbonell, J. G., "A Natural Language Processing Tutorial", Tech. report, Carnegie-Mellon University, Computer Science Department, 1983.

7. Carbonell, J. G. and Hayes, P. J., "Coping with Extragrammaticality", *Proceedings of COLING-84*, Stanford, CA., June 1984.

8. Hobbs, J. R., Stickel, M., Appelt, D. and Martin, P., "Interpretation as Abduction", Tech. report Technical Note 499, SRI International, 1990.

9. Charniak, E., "Motivation Analysis, Abductive Unification, and Nonmonotonic Equality", *Artificial Intelligence*, Vol. 34(3), 1988.

10. Wilensky, R., *Planning and Understanding*, Addison Wesley, Reading, MA, 1983.