USING PRAGMATIC AND SEMANTIC KNOWLEDGE TO CORRECT PARSING OF SPOKEN LANGUAGE UTTERANCES

Sheryl Young and Michael Matessa

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. 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 six non-overlapping test sets, each evaluated for both speech and speech transcription processing. These test sets evaluate the systems ability to enhance performance in both highly restricted and completely unrestricted input data. Further, some test sets capitalize upon the unique linguistic features of spontaneous speech. These evaluations illustrate that the decrease in incorrect interpretations and total error rate resulting from SOUL's postprocessing are most pronounced in unrestricted transcript data and all forms of speech data, as opposed to carefully constrained test sets. For example, in processing transcription data incorrect interpretations are reduced by 53% in constrained sets, as opposed to 81% in unrestricted sets. In other words, the more difficult the processing and/or interpretation, the more there was to be gained by using extensive reasoning abilities.

1. OVERVIEW

The DARPA speech and natural language community has recently adopted the domain of air travel information for its spoken language systems. The domain is called 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 problems with predefined goals and preferences. To perform their task, speakers must verbally query an air travel database. The task is designed to elicit natural, unedited, spontaneous speech. Speakers are NOT restricted to complete, unambiguous, answerable, well formed or syntactically correct utterances. They can ask for any information, regardless of whether the request is reasonable or answerable by the information contained in the database. Training and test data are collected by recording both subject utterances and database responses as speakers perform these verbal tasks.

The SOUL system, described in this paper, is designed to process spontaneous speech, inclusive of all its dysfluencies, on-line corrections and edits as well as the ambiguous and unanswerable utterances commonly found in the spontaneous speech of a naive system user.

1.1. SOUL OVERVIEW

SOUL is a general purpose, in-depth knowledge based reasoning system. It relies primarily upon pragmatic and semantic knowledge which is organized into a multi-layered frame base system of hierarchies.[1] It reasons by using abductive reasoning and constraint satisfaction techniques. The SOUL system is designed to perform thorough analyses of individual, spoken utterances. The in-depth reasoning component augments the basic semantic and syntactic information which is rapidly obtained by caseframe parsing of speech, by performing a more fine grained analysis of the hypothesized speech input. The system operates in an opportunistic manner. It is only called upon when there exists reasonable uncertainty in how to interpret the input. Furthermore, when hooked into the dialog module (MINDS-II), it uses additional knowledge of inferred speaker intentions (goals & plans) to assist in deriving a complete and accurate interpretation of the spoken input.

1.1.1. THE CMU ATIS SYSTEM

We designed the SOUL system to enhance the interpretation accuracy of individual utterances input to the CMU ATIS system. The CMU system has been configured in two separate manners. The first is a thoroughly integrated system as presented in the original MINDS papers [1, 2, 3, 4]. There, the parser and goal and plan module determined what words the speech recognizer could try to recognize from the incoming speech signal. The configuration currently in use for the ATIS system is slightly less expectation driven, the higher level knowledge sources only directing the recognizer when semantically un-meaningful, inconsistent or poorly recognized information is output from the recognizer.

The CMU ATIS System is composed of four interacting modules; a speech recognizer (SPHINX), a robust case-frame parser designed to process spontaneous speech (PHOENIX), a pragmatic and semantic module for individual utterances (SOUL) and the dialog component which hypothesizes and tracks goals and plans (MINDS-II). SOUL interacts with all the other modules. The recognizer uses a bi-gram grammar to produce a single

[1]The knowledge can best be viewed as a set of planes, each containing multiple hierarchies, where each plane contains information about entities of similar granularity. For example, objects and their attributes are contained within a single plane, where as actions and events are contained within a second plane. This representation lessens the inconsistencies normally found in large reasoning systems.

output string which is fed to the speech case-frame parser. The parser generates a phrase matrix by matching all meaningful phrases in the recognized string. It then constructs a beam of plausible interpretations. Using robust slot filling heuristics, all possible case-frame slots that can be filled by any recognized phrase 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. If an unambiguous interpretation that accounts for all input is discovered, its passed directly to the dialog module. If not, SOUL is called first.

Input to SOUL is the string produced by the recognizer, the phrase matrix and the beam of plausible interpretations. Output from SOUL is either an indication that the input is unanswerable, (e.g. No information in database on BOULDER), a single interpretation composed of corrections, deletions, and all forms of modifications to the instantiated case-frame, or a message that an incomplete interpretation exists.

In the version of the system responsible for the reported results, SOUL could tell what the parser mis-understood and suggest plausible hypotheses regarding the user's intentions. In the version which exists at time of writing, SOUL can re-process questionable input and try again to derive an accurate and complete interpretation. Reprocessing involves sending any questionably recognized input back to the recognizer along with a set of possible words matches and a grammar to replace the generic bi-gram grammar. Finally, if an interpretation can be generated, it is passed to both the dialog module and the database. Database output is presented on a computer screen to the user.

1.2. RESEARCH ISSUES

The rationale for developing SOUL was prompted by the unique characteristics of spontaneous utterances which violate many of the underlying assumptions of typed or well formed textual input and make spontaneous speech much more difficult to accurately interpret. Consider the following sources of error and uncertainty: mis-speaking, inaccurate speech recognition, incomplete utterances, unmeaningful utterances, ambiguous input, ungramamticality, poorly formed input, speaker generated noise, stuttering and mid-utterance corrections and edits. In other words, there are many sources of ambiguity in a recognized speech signal. This results in large regions of un-accounted-for speech, multiple, competing interpretations and seemingly incomplete or un-meaningful interpretations. 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 and in-depth analyses 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.

Second, we wanted to determine under what input conditions the use of the additional knowledge and reasoning capabilities would significantly enhance performance. For example, would the addition of SOUL allow us to process unanswerable utterances, where speakers request information not in the database and outside the definition of system capabilities? Will we be able to detect those utterances which cannot be answered due to unavailable contextual information, which contain semantically ambiguous input, or regions of speech containing word substitutions and recognition errors? Finally, would the system enhance overall performance only under the more difficult or more ambiguous conditions, or would performance increase irrespective of the type of input?

To evaluate these questions, we assessed performance of our case-frame speech parser, PHOENIX, both with and without SOUL on a series of independent test sets. The test sets vary with respect to two variables. The first variable was input mode. Performance was assessed under three conditions: spontaneous speech transcriptions, spontaneous speech, and spontaneous speech containing verbal edits and corrections. Our second variable was complexity of input. Here we contrast constrained, answerable and unambiguous input with unconstrained, uncensored input. The result is an assessment of performance of the system with and without SOUL under six separate data conditions.

The remainder of this paper describes the heuristics and algorithms used by SOUL to correct, reject and/or clarify the outputs of the PHOENIX case-frame parser in the ATIS domain and performance evaluations. 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 performance evaluations which contrast the performance of the PHOENIX case-frame parser with and without the SOUL postprocessor.

2. LINGUISTIC PHENOMENA EXAMINED

SOUL was developed to cope with the specific demands of spontaneous speech sentence processing . Further, we hoped its in-depth reasoning capabilities would enhance system accuracy by augmenting the basic, rapid pattern matching techniques used in our caseframe parser, PHOENIX. It followed that we would need to know which linguistic phenomena were problematic and the prevalence of these phenomena. Therefore, we took a set of 472 training utterances and analyzed them in terms of which posed problems for the caseframe parsing system and / or which contained unique linguistic attributes not found in typed input. An evaluation of the performance of the June 1990 version of the 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:[2] (Note: underlined information not in database)

Unanswerable queries, no information in database, or illegal

[2]It should be noted that two of our categories would not be expected to be problematic for most well developed typed input systems. These are superlatives and comparatives and quantified yes/no's.

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?

While many of the phenomena in the above list would prove problematic for most natural language systems, particularly speech systems, there is another whole class of phenomena, unique to speech which merit discussion on their own right. These are the characteristics of spontaneous speech.

2.1. SPONTANEOUS SPEECH CHARACTERISTICS

Spontaneous speech contains many forms of mid-utterance repetitions and corrections, also called verbal editing. Consider the following example, *Show me the uh how what does V U 1 stand for*. To accurately process speech with verbal edits, you must purge the utterance of the unintended words. Spontaneous speech is filled with both silent and voiced pauses, *e.g. uh, uhm* and both ungrammatical and ill-formed utterances. Voiced pauses are a problem because unless specifically trained otherwise, a speech recognizer will try to map the sounds into non-existent words. In doing so, as in many substitution errors, prior and later words are misrecognized in an attempt to match the noise sound with a real word. An example of an ill-formed utterance is, *I need to uh sometime after twelve Dallas time uh leave uh be at Denver uh before two but closer to twelve their time if possible*. Other sources of noise in addition to filled pause include partial partial words, *e.g. Which of ah f-fl- flights ....* Corrections *e.g. Show me uh oh excuse me which cost less than 200 oh make that 250 dollars.* as well as incomplete and multiple requests occurring in the same segment of speech are also common. For example, *Which of the* and *Show all flights that land before 5:30 p.m. and what ground transportation is available*. Finally, as discussed above under "unanswerable queries", unconstrained dialog contains many different forms of out of domain utterances. These include meta-level comments stating the speaker's intentions, goals, goal conflicts and goal progress *e.g. OK, now I've got the first leg finished ...*; requests for help *e.g. What do I have to do to reserve a limousine?, How do I read these tables?* Further forms of out-of-domain requests include asking for information not included in the database as well as requests for the database to perform actions which it is not able to perform.

To further complicate these spontaneously occurring phenomena, one must also consider that speech recognition itself is inexact, with accuracy decreasing as the number of allowable words and sounds increases.

3. 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).

3.1. 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. Hence, the process can be described as using the representations of identified knowledge base entries and lookuing to see the ways in which they can be most specifically related to one another, while maintaining consistency among the entire set of entries.

To illustrate, consider the following utterance: *Does B stand for business class*. The case-frame parser instantiates the following sequence of concepts or cases:

```
[list] DOES
[CLASS]
  [class] B
[describe_code]
  [class] B STAND FOR
[CLASS]
  FOR [class_type] BUSINESS CLASS
[describe_column]
  [column_name] CLASS
```

The knowledge base is faced with three basic concepts: the letter B, stand-for, and Business-class. Although not told by the parser, the knowledge base knows that "B" could be an abbreviation for Breakfast or Coach-class, part of a flight number identifying the airline carrier, part of the call letters of an aircraft, or one of the letters composing the name of a person. 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 it is 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, SOUL sees that Business-class is not equivalent to Breakfast or Coach-class, and signals PHOENIX that the utterance is an equivalence query with a negative answer.

This same mechanism also determines what portions of an utterance, or what words and phrases can form meaningful units. Here, reliance upon the knowledge base constraints for slot fillers and relations among slot fillers is most important. To illustrate, consider the utterance *What are arrival and departure times for flight ten fourteen from Philadelphia to Dallas?* The parser would instantiate the following cases:

```
[explain] WHAT ARE
[list] WHAT ARE
[FLIGHT_NUM] FOR
  [flight_num]
    FLIGHT [number] TEN FOURTEEN
[ARRIVE_LOC] TO
  [arrive_loc] [city] DALLAS
[DEPART_LOC] FROM
  [depart_loc] [city] PHILADELPHIA
[city_airport] FROM
  [airport_name] PHILADELPHIA TO
  [city] DALLAS
```

The best interpretation of the parser would be "Explain 1014 and PHL and DFW" with alternative interpretations of "List-flight 1014 from PHL to DFW" and "Show-distance from PHL to DFW". All phrases matched and all interpretations are input to SOUL along with the original utterance. SOUL first notices information from the utterance missing in the interpretations and determines that "arrival" and "departure" and "times" are all concepts of List-flight. Finding no other information that conflicts with the interpretation of List-flight being the subject of the utterance, SOUL signals PHOENIX to change its interpretation from Explain to List-flight, and since arrival and departure times are shown with List-flight, does not suggest any additional changes.

The abduction method also detects critical information missing from the parse, even though nothing may be known about that information. For example, for the utterance ...*from Atlanta to Denver arriving in Denver before lunch*, the parser returned "List-flight ATL to DEN". SOUL signalled with "Stop: bad-time LUNCH" because it knew some time designation should be in that location, but didn't know that lunch is usually served around noon.

3.2. 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 limousines. 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.

3.3. 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.

3.4. SPEECH PROCESSING

The task of processing PHOENIX's parsings of speech is more difficult that of processing its transcript parsings. With the exceptions of unknown lexical items and mis-speaking, a speech transcript gives a near-perfect copy of what the speaker has said. While out of domain and ill-formed requests may occur, transcript processing is still much easier that speech processing. The output from a speech recognizer is subject to several additional types of errors, including mis-recognizing words as other words (substitutions), mis-recognizing noise as words, and adding and deleting words. To deal with unknown word strings in the case of transcript processing as well as the sources of error in speech described above, we developed special capabilities .

There are three possible outcomes when dealing with unrecognized / mis-matched speech input. In the first case, the recognizer can substitute with a word or set of words which are semantically meaningful in the context of the surrounding input. The second and third cases are when there is a sequence of one of more mis-matched words (or unknown words in a transcript). In this case, we must detect that there is an unaccounted-for region, and then hypothesize which known entities and actions in the utterance the region of text could modify it. Next, we determine reasonable semantic values the region could take. Occasionally, the region is limited enough to determine what the mis-matched word really were. More likely, however, we cannot. In these cases, we have the options of sending the information back to the recognizer or telling the speaker the region of speech is not understood. When we feed back to the recognition system, the inconsistent region is reprocessed along with a grammar which represents methods of expressing the sum of our hypothesized semantic contents. Examples of all three cases are shown below.

This example illustrates why one cannot always detect the presence of a recognition error. Of course, if the system does not recognize the existence of an error, there is no way it can infer the correct words. This is the case when the recognizer output a word substitution that is semantically meaningful in the context of the surrounding words.

`Speaker:`	*Show fare for those.*
`SPHINX:`	***Show flights from Dallas.***
`PHOENIX:`	**list flights from DFW**
`SOUL:`	**OK**

Examples of errors that are detected by SOUL but not corrected include both cases of incompleteness, e.g. a variable introduced but not bound, and regions of meaningless words which are inconsistent with surrounding context.

`Speaker:`	*Let me see the flights from Oakland* *to Baltimore arriving before noon.*
`SPHINX:`	***Show me see the flights from Oakland*** ***to Baltimore each arriving*** ***before name.***
`PHOENIX:`	**list flights from OAK to BWI**
`SOUL:`	**Stop: bad-time NAME**

And some missing or altered words can ruin the correct parsing of PHOENIX but still allow SOUL to determine the meaning of the utterance by context.

`Speaker:`	*Please list only flights from Dallas* *to San Francisco on Delta airlines.*
`SPHINX:`	***Please list only flights from Dallas*** ***Francisco on Delta airlines.***
`PHOENIX:`	**list flights from DFW**
`SOUL:`	**Add: SFO -> destination city**

`Speaker:`	*Show me departing flights from* *San Francisco* *to Dallas arriving in Dallas* *by one pm.*
`SPHINX:`	***Show me departing flights from*** ***San Francisco*** ***to Dallas arriving in Dallas*** ***my one pm.***
`PHOENIX:`	**list flights from SFO or OAK** **to DFW departing at 1300**
`SOUL:`	**Add: 0-1300 -> arrive time;** **Remove: depart time**

As these examples show, the speech component of SOUL relies heavily on the conceptual relationships of the knowledge base to determine context.

Along with recognition errors, spontaneous speech introduces the added difficulty of verbal editing. Verbal editing consists of repetitions and corrections. Repetition of voiced pauses or noise words, e.g. *uh*, *uhm*, is processed by PHOENIX, which filters the words out before parsing. Repetition of meaningful words can be thought of as correcting an old word with the identical new one. The caseframe parsing of PHOENIX allows corrections to be made by either replacing or adding to the value of a frame slot with a semantically similar value. While this frequently suffices, it is difficult to apriori determine whether one should overwrite or replace a slot value without the use of additional semantic and pragmatic knowledge. SOUL uses the context of the whole utterance to insure that the correction makes sense.

For example, in the utterance ...*show me the prices uh for US airlines flight let's see it should be what flight five twenty from Boston to Pittsburgh no excuse me flight one forty eight from Boston to Pittsburgh*, PHOENIX would add *flight 148* to *flight 520* in the Flight slot, but SOUL would delete *flight 520* because there are words between the two concepts indicating a correction is being made.

3.5. SOUL OUTPUT

Earlier, we said that SOUL can correct inaccurate interpretations often without sending the regions back to the recognizer for reprocessing. Here, we specify the types of instructions and codes sent back to the PHOENIX parser for incorporation into the database query.

- When there is missing, critical information, 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 misconceptions. (e.g. *Show all flights from Dallas to Fort Worth*).
- Finally, when two unrelated 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 Philadelphia to Boston and how far Boston is from Cape Cod* would be divided up where as *Show all flights from Philadelphia 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.

4. EXPERIMENTAL RESULTS

In this section we evaluate the SOUL system on a variety of both verbatim speech transcripts and spoken data. Our test sets are composed of completely unrestricted input as well as restricted, official DARPA evaluation criteria. Furthermore, we present evaluations of spontaneous speech which contains mid-utterances edits and corrections.

Our rationale for presenting the varied evaluations is two-fold. First, SOUL was designed to deal with more complicated phenomena, normally found in completely unrestricted, spontaneous speech. However, the prevalence and importance of these features has not previously been assessed. Hence, we will be able to see the type of difficulty imposed and the efficacy of our solutions. Further, we will be able to asses our hypothesis that the SOUL system will enhance performance more significantly in unrestricted or more difficult processing situations. Our reason for evaluating the system on restricted input is that currently, the official, required DARPA evaluations include only highly constrained utterances. These have been divided into two types of utterances. Type A input includes only well-formed, unambiguous utterances which are answerable using the information in the database and do not rely upon any prior context. Type D1 utterances are pairs of Type A utterances where the second utterance must be answered by remembering information expressed in the first utterance in the pair. Type D1 pairs are randomly constructed and do not normally follow one another in a natural dialog. 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 all offical DARPA test material.

4.1. PILOT STUDIES

The current implementation of SOUL was trained on the first two-thirds (472 utterances) of the ATIS0 training data available in June 1990. The final third was reserved for testing along with the official June 1990 DARPA test data. There is a significant difference between these two data sources. The DARPA data is highly constrained, containing only Type A utterances, as described above. The 232 additional utterances were completely unrestricted. These were divided into two test sets. One test set contained all utterances which could only be interpreted in the context of the surrounding dialog, the rest were interpretable without the benefit of dialog. Specifically, Set 1 contained the 94

Class A from the official June 90 ATIS-0 test set. Sets 2 and 3 both used the unrestricted utterances provided in June 1990 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 226 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 25 context-dependent utterances.

Results of the three evaluations on transcript data and speech data, comparing the performance of PHOENIX alone and PHOENIX plus SOUL are given in the tables below. 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 improvements 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 queries, derive multiple interpretations for ambiguous input, to interpret ill-formed and ungrammatical input and to correct inaccurate output from the PHOENIX parser. Finally, results from Test Set 3 indicate SOUL's proficiency in determining when PHOENIX has enough information to correctly parse a context dependent utterance. However, it should be noted that the Test Set 3 results are not representative of PHOENIX performance. PHOENIX is designed to process context dependent utterances only when using context.

Results from the Three Transcript Test Sets					
Test Set	System	% Correct	% Incorrect	% No Answer	Total Error
Test Set 1	PHOENIX	79.79	20.21	0.00	20.21
	PHOENIX + SOUL	79.79	9.57	10.64	20.21
Test Set 2	PHOENIX	67.26	32.30	0.44	32.74
	PHOENIX + SOUL	73.01	6.19	20.80	26.99
Test Set 3	PHOENIX	0.00	100.00	0.00	100.00
	PHOENIX + SOUL	52.00	36.00	12.00	48.00

Results from the Three Speech Test Sets					
Test Set	System	% Correct	% Incorrect	% No Answer	Total Error
Test Set 1	PHOENIX	55.32	43.62	1.06	44.68
	PHOENIX + SOUL	55.32	23.40	21.28	44.68
Test Set 2	PHOENIX	48.67	49.56	1.77	51.33
	PHOENIX + SOUL	53.98	23.89	22.12	46.01
Test Set 3	PHOENIX	0.00	96.00	4.00	100.00
	PHOENIX + SOUL	40.00	36.00	24.00	60.00

The results from the speech data show the effect of word recognition errors on the three test sets. There is an overall decline of performance due to speech corruption, but the results for PHOENIX alone in the third set remain roughly the same. This was due to one utterance becoming too corrupted to parse and to PHOENIX still not having enough information to give correct answers for the others.

4.2. FEBRUARY 1991 DARPA DATA

The February 1991 test set contained 148 isolated utterances, 38 utterances which are interpreted using limited context, or context provided by the preceding query and its answer, and 11 utterances that contain verbal editing. The 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 11 "Optional" utterances contain both repetitions and corrections. The results of the evaluation are presented below.

ATIS Transcript Data Results					
Test Set	System	% Correct	% Incorrect	% No Answer	Total Error
TYPE A	PHOENIX	82.43	17.57	0.00	17.57
	PHOENIX + SOUL	91.22	4.73	4.05	8.78
TYPE D1	PHOENIX	81.58	18.42	0.00	18.42
	PHOENIX + SOUL	81.58	9.21	9.21	18.42
OPTIONAL	PHOENIX	81.82	18.18	0.00	18.18
	PHOENIX + SOUL	81.82	0.00	18.18	18.18

ATIS Speech Data Results					
Test Set	System	% Correct	% Incorrect	% No Answer	Total Error
TYPE A	PHOENIX	59.46	39.86	0.68	40.54
	PHOENIX + SOUL	61.49	17.57	20.95	38.51
TYPE D1	PHOENIX	67.11	32.89	0.00	32.89
	PHOENIX + SOUL	68.42	17.11	14.47	31.58
OPTIONAL	PHOENIX	54.55	45.45	0.00	45.45
	PHOENIX + SOUL	54.55	18.18	27.27	45.45

An analysis of the results indicates that SOUL is responsible for a slight decrease in overall error rate. The results also indicate that SOUL is reasonably good at detecting and flagging inaccurate interpretations and stopping PHOENIX from making incorrect interpretations. Again in the speech condition there is an overall decline of performance due to word corruption.

5. 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 that are unanswerable due to unavailable contextual information. Results of Test Set 3 indicate reasonable detection abilities (52%).

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.

6. REFERENCES

1. 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. .

2. Young, S. R. and Ward, W. H., ''Towards Habitable Systems: Use of World Knowledge to Dynamically Constrain Speech Recognition'', *The Second Symposium on Advanced Man Machine Interface Through Spoken Language*, 1988.

3. 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.

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.