The Delphi Natural Language Understanding System

Madeleine Bates, Robert Bobrow, Robert Ingria and David Stallard
BBN Systems and Technologies, Inc.
70 Fawcett St.
Cambridge, MA 02138

Abstract

This paper presents Delphi, the natural language component of the BBN Spoken Language System. Delphi is a domain-independent natural language question answering system that is solidly based on linguistic principles, yet which is also robust to ungrammatical input. It includes a domain-independent, broad-coverage grammar of English. Analysis components include an agenda-based best-first parser and a fallback component for partial understanding that works by fragment combination. Delphi has been formally evaluated in the ARPA Spoken Language program's ATIS (Airline Travel Information System) domain, and has performed well. Delphi has also been ported to a spoken language demonstration system in an Air Force Resource Management domain. We discuss results of the evaluation as well as the porting process.

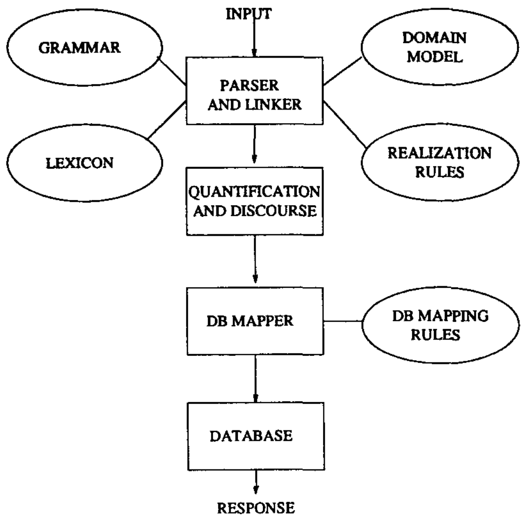

Figure 1: System Diagram

1 Introduction

Delphi is a natural language understanding system based on general linguistic principles which is adaptable to any question-answering domain. It incorporates a number of domain-independent knowledge bases, including a general, broad-coverage grammar of English with a powerful and flexible handling of complementation. Unlike most other linguistically motivated systems, however, Delphi is also highly robust, allowing for partial understanding when an input is ungrammatical, disfluent, or not properly transcribed by a speech recognizer. Thus, Delphi can be used for a spoken language application as readily as for a written one. Furthermore, Delphi's partial understanding component, called the Semantic Linker, is driven off the same system of semantic rules as Delphi's regular best-first parser. Building a robust application therefore requires no additional effort.

There are several components of the system, which is diagrammed in Figure 1. First are the parser and Semantic Linker, which output an intermediate representation we call a "semantic graph". The semantic graph is passed to a quantification stage which produces a fully scoped logical form from it. The logical form is then passed to the discourse stage, which resolves pronominal references and performs other types of task-dependent constraint resolution to produce the final logical form. The final logical form is then passed to the backend translator, and then to the application system which produces the response. Several knowledge bases are employed by these analysis components, including grammar, "realization rules" and the domain model, which represents the set of classes and binary relations of the given application domain.

Delphi differs from most other linguistically motivated systems in the role that is played by syntax. The primary function of Delphi's parser and syntactic knowledge bases is not to produce a parse tree, but rather to constrain the search for an appropriate semantic graph interpretation of the utterance. Semantic graphs are produced not by rule-to-rule compositionality, but by what might be called "relation-to-relation" compositionality – the association of grammatical relations in the syntactic structure with semantic relations in the semantic graph.

This more incremental view of the syntax/semantics interface has three crucial advantages. First, there is much more flexibility with respect to ordering and optionality of constituents. Second, because relation-to-relation translations are simple, the task of porting the system is greatly simplified. Third and finally, partial or fragmentary analyses can be represented, and a complete semantic graph interpretation for the utterance produced even when a complete syntactic analyses is not available.

In the remainder of the paper, we describe Delphi's main processing components, representational formalisms, and knowledge bases.

2 Grammar And The Syntax/Semantics Interface

The Delphi grammar is a broad coverage, domain independent grammar of English written in a version of the Definite Clause Grammar formalism (Pereira and Warren, 1980) that has been extended to include labeling of right-hand side elements with the grammatical relations they bear to the head of the construction. An example is:

```
(S ?arg ?mood)
->
subject: (NP ?arg ?mood etc.)
head:    (VP ?agr ?mood etc.)
```

In this rule, there is a head VP and an NP which bears the SUBJECT relation to it. Other grammatical relations include the familar DIRECT-OBJECT and INDIRECT-OBJECT as well as the prepositions, such as TO, FROM, WITH and so on.

Annotating sub-constituents with grammatical relations regularizes the syntactic structure with respect to particular grammatical rules, and allows a "relation-to-relation" form of compositionality, as opposed to the more traditional "rule-to-rule" version that is exemplified by such systems as Gemini (Dowding et al, 1993) and the Core Language Engine (Alshawi, 1992). In relation-to-relation compositionality, each grammatical relation in the syntactic structure corresponds to a semantic relation in a parallel semantic structure we call a "semantic graph". The terminal nodes of the semantic graph are the word meanings, corresponding to the lexical heads of syntactic structure.

An example of a semantic graph, representing the meaning of "What flights fly from Boston to Denver", may be seen in Figure 2. The semantic graph is not a fully quantified formula; rather it may be thought of as a form of predicate-argument representation, with quantifiers in place, from which a fully quantified formula can be generated. The allowed class and relation labels come from the domain model.

This view of the syntax/semantics interface has marked advantages. For one thing, because the syntactic/semantic structure is built up one argument at a time, it becomes much easier to accomodate such phenomena as order-variation and optionality of arguments that are difficult for other approaches

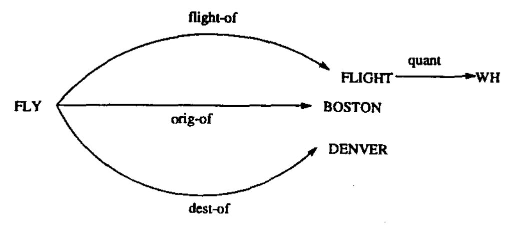

Figure 2: Semantic Graph

The importance of this feature may be seen in the examples of argument order-variation and optionality that abound in real data. Consider the following from the ATIS domain, in which complements can vary freely in order:

What flights fly from Boston to Denver?
What flights fly to Denver from Boston?

or be separated from the head by a modifier typically regarded as an adjunct:

What flights fly at 3 pm from Boston to Denver?

In some cases, modifiers can be omitted, as in:

What flights fly from Boston?
What flights fly to Denver?

and sometimes the omission of an argument can have anaphoric consequences, as in:

What restrictions apply?

which cannot be felicitously uttered except in a context where there is something in the discourse that a restriction could "apply" to.

Conventional approaches to subcategorization, such as Definite Clause Grammar (Pereira and Warren, 1980), Categorial Grammar (Ades and Steedman, 1982), PATR-II (Shieber, 1986), and lexicalized TAG (Schabes et al, 1988) all deal with complementation by including in one form or another a notion of "subcategorization frame" that specifies a sequence of complement phrases and constraints on them. Handling all the possible variations in complement distribution in such formalisms inevitably leads to an explosion in the number of such frames, and a correspondingly more difficult task in porting to a new domain.

In our approach, on the other hand, it becomes possible to view subcategorization of a lexical item as a set of constraints on the outgoing arcs of its semantic graph node. Different types of constraints – order of arguments, optionality of arguments, semantic-class constraints and semantic effects of arguments – can all be represented separately, instead of enumerating all possible argument sequences in a set of alternative subcategorization frames.

Subcategorization constraints in Delphi are encoded in lexical entries using a structure called a "map" (Stallard and Bobrow, 1991). Below is part of the lexical entry for "fly" in the ATIS domain:

```
FLY
subject: FLIGHT-OF
to:      DEST-OF
from:    ORIG-OF
completion: (and (filled flight-of)
                 (or (filled dest-of)
                     (filled orig-of))
```

Map entries have "translation", "realization" and "completion" components. The translation part of this entry specifies that the lexical head "fly" is to correspond to a semantic-graph node labeled with event-class FLY. The realization part of the entry specifies what grammatical relations the lexical item takes, and what semantic relations these correspond to, or "realize", in the semantic graph. Here, the entry specifies that "fly" takes SUBJECT, TO, and FROM complements, and that these grammatical relations correspond to the semantic relations FLIGHT-OF, DEST-OF, and ORIG-OF respectively. Semantic selectional restrictions in these argument positions – that the filler of DEST-OF be a city, for example – are implicit from the declarations of the relations in the domain model.

The "completion" part of the entry specifies what outgoing arcs are *required* for the node. Here, the entry requires that the FLIGHT-OF role be filled, and that either the DEST-OF or ORIG-OF roles be filled (forbidding the intransitive "the flight flies"). More complex optionality cases are encoded with other completion predicates. For example, the case where an anaphor must be present ("What restrictions apply") is encoded by the predicate FILLED-OR-ANAPHOR.

Some realization rules are tied to semantic classes rather than lexical translations, and require for their application only that semantic class restrictions implicit from the domain and range of the realized relation be satisfied. Typical examples are the rules governing noun modifier meanings, such as "Delta flights", "Delta's flights", "the flights on/aboard Delta". These would all be handled by the global realization rule:

{NOM-COMP POSS ABOARD ON ...}
$\longrightarrow$
AIRLINE-OF

Determining what semantic relation a given grammatical relation instance corresponds to is most generally viewed as a form of goal-solving in Delphi, in which a chain of rules can be invoked. For example, syntactic constructions such as "X with Y", "X has Y" and "X's Y" are interpreted by first appealing to a rule mapping them to a pseudo-relation called GENERALIZED-POSSESSION, and then seeking a realization for it that is compatible with the classes of X and Y. This avoids having to write three different versions of the same realization rule.

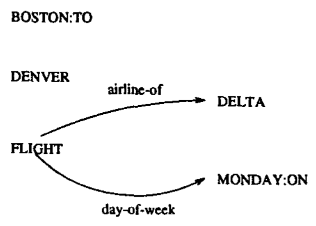

Figure 3: Fragment Graphs

An important advantage of the realization rule formulation, apart from its its power and flexibility, is its simplicity. Realization rules are very simple to write, and make maximal use both of knowledge about the domain and general knowledge of language.

3 Ill-Formedness Handling: The Semantic Linker

When an utterance cannot be parsed with Delphi's best-first parser (Bobrow, 1991) – either because it is ill-formed, mis-recognized by the speech system, or simply because it is outside the coverage of the grammar – it can still be partially understood by the system, often well enough to give the correct response. The component responsible for partial understanding in the Delphi system is called the Semantic Linker (Stallard and Bobrow, 1993).

After a parse fails there is a set of fragmentary constituents left over in the chart, corresponding to a set of semantic graphs. The Semantic Linker seeks to connect these sub-graphs into a single connected one by adding links between nodes in the different sub-graphs.

At top-level, this is the same thing that the parser and grammar do. The difference is that the parser and grammar have an idea of what the grammatical relationship between constituents is, based on requirements of their proximity in the string and other syntactic evidence. The Semantic Linker does not have these requirements, being a looser form of combination that can ignore fragment order and skip over intervening, unanalyzable material with ease.

Although it is a very different algorithm, the Semantic Linker uses the same set of realization rules that drives the regular parser. Using the realization rules, the Linker determines for each pair of nodes in different semantic graphs the set of all links which can connect them. It then uses an A* search to find the most plausible set of links which produce a complete graph.

Suppose for example, we have the three fragments "to Boston", "Denver" and "Delta flights on Monday". Then the three corresponding sub-graphs are those shown in Figure 3 where a PP is treated as its NP object with the preposition as a tag. For this set of fragmentary sub-graphs, the possible links are:

```
1a.  FLIGHTS1--- DEST-OF -> BOSTON:TO
```

```
1b.  FLIGHTS1--- ORIG-OF -> BOSTON:TO

2a.  FLIGHTS1--- DEST-OF -> DENVER
2b.  FLIGHTS1--- ORIG-OF -> DENVER

3a.  DENVER--- NEARBY-TO -> BOSTON:TO
```

where the links are grouped together in a ordered list according to the fragment-pairs they connect.

The plausibility of a given link is a function of a number of different features, including penalities from assumptions made in its computation (e.g. that a given preposition can be ignored or assumed) and empirically determined probabilities for the given link (e.g. that given an AIRLINE and a FLIGHT they are most probably linked by the relation AIRLINE-OF).

The semantic linker may also "hallucinate" a new node to bridge two fragments between whom no links can otherwise be computed. For example, for the utterance "from Boston to Denver", which has no explicit FLIGHT-object, a FLIGHT node can be inserted between the fragments to make sense of the utterance.

Because the Semantic Linker uses the same set of realization rules as the rest of the system, when the system is ported to a new domain the Semantic Linker can be used immediately – a distinct advantage over some other approaches to fallback understanding, such as (Stallard and Bobrow, 1992) or (Jackson et al, 1991).

In formal experiments (as we discuss subsequently) the Semantic Linker has been show to dramatically improve Delphi's performance.

4 Quantification

The quantifier scoping module in Delphi takes a semantic graph and produces a fully-scoped expression in the logical language FMRL. The basic strategy for quantifier scoping is a descendant of that used in the LUNAR system (Woods et al, 1978). This is made possible by the use of the semantic graph as a common underlying representation for both the grammatical and ill-formed parts of fragmentary utterances. Delphi's scoping module traps quantifiers from relative clauses, makes the quantifiers from PPs etc. outscope the NP quantifier, and resolves the scope of quantifiers from parallel constituents in terms of left-to-right order in the input. These general rules are modified to take into account differing strengths of quantifiers such as EACH.

Left-to-right ordering and syntactic structure for grammatical portions of the utterance are recovered from the semantic graph by backpointers to the lexical items and grammatical relations from which the graph was produced. Links established by the semantic linker are treated by the quantification mechanism as if the constituency is indeterminate, so that only left-to-right scoping rules and individual quantifier preferences take effect.

The resulting mechanism is robust, and quantificational scoping has been an insignificant source of error in the official ARPA blind-test evaluations of the ATIS system. More complex strategies have been proposed and implemented in the last two decades, and could in principle be modified to work with ill-formed input, but the simple and robust LUNAR approach handles essentially all the phenomena seen in the tens of thousands of sentences of ATIS training collected during experiments with non-linguist users.

5 Discourse

The discourse mechanism of Delphi consists of several components: resolution of local ambiguities, pronominal and deictic antecedent resolution, ellipsis handling and discourse constraint propagation.

The most common case of local ambiguity in the ATIS domain involves temporal phrases as in "the nine o'clock flight". The resolution mechanism searches both for linguistic information in the current and previous sentences, as well as properties of entities in previous answers, to resolve whether "nine o'clock" is AM or PM.

The pronoun/deictic resolution mechanism used in Delphi makes use of locally expressed or implied semantic constraints to search through a set of candidate antecedents. The current mechanism ignores syntactic number as a cue, because empirically in the ATIS corpus (and we suspect in other spontaneous speech applications) it is often in error. A simple-minded focus component is used, primarily based on recency, and secondarily based on grammatical relations within an utterance. Because of the strength of semantic cues and the prevalence of ill-formed input, the use of syntactic cues for focus is limited.

The interpretation of later sentences often must include information from previous sentences, without explicit linguistic cues. This is especially true in "design dialogues", where the goal is to find a description of a set of objects that will meet some set of implicit or explicit constraints. Consider for example the following discourse from the ATIS domain.

Show Delta flights from Boston to Dallas tomorrow.
Can I leave in the morning?
Is there a nonstop flight?
Show me the American flights.
I want to go from Dallas to Chicago on Wednesday

Note that the constraints of prior sentences (such as on airline, origin, destination etc.) are implicit for subsequent sentences unless contradicted by information in the current sentence (e.g. "American" overrides the "Delta" from the first sentence) or until there is evidence that a new problem is being solved (the new origin and destination in the last sentence indicates that all previous constraints can be dropped). Delphi has a "context tracker" that maintains a stack of the constraints from previous utterances, and has a set of rules for when constraints are to be modified or deleted before being merged with the current sentence.

Finally, we handle ellipsis as a special case of semantic linking. If we have the two utterances:

Show me the meals on the morning flight.
on American at 12:30

We can treat these as if they were one run-on ill-formed input and link "American" to "flight", and replace "morning" with "12:30", using a minor variant of the Semantic Linker linker which allows for later constraints to overwrite earlier ones of the same type. This strategy has been very effective, and covers a large class of elliptical constructions.

6 Backend Mapping

In order to get a response to a user query, the complete FMRL interpretation of an utterance must be translated to an expression of a target query language which can be evaluated directly against the tabular database to retrieve the answer.

A key step is bridging the gap in conceptual vocabulary between the two representations. For example, the FMRL interpretation of the query "How many flights on Delta serve meals" has one-place predicates like FLIGHT and AIRLINE, and two-place predicates like AIRLINE-OF and MEAL-OF. The database for the ATIS domain, on the other hand, only has a single table FLIGHT with fields containing airline and meal information. Delphi bridges this gap between representations with a system of local mapping rules which translate the one- and two-place predicates of the FMRL into expressions of a relational algebra target language which retrieve the extensions of these predicates.

Sometimes, however, some combination of FMRL predicates has a correspondence in the database but the individual predicates themselves do not. For example, in the database for the SPLINT domain a table relating aircraft-types to their physical characteristics has a field for the *number* of engines the aircraft has, but no representation for the engines themselves. If we now ask "How many engines does an F-16 have?", there is no local translation of the FMRL predicate ENGINE.

To deal with this, Delphi has a system of global transformations that are applied first, rewriting subsets of the FMRL clauses to a form that can be handled with local translation. The rule that handles this example is:

```
(is-a :e engine number)
(aircraft-engine-of :a :e)
→
(is-a *count* number)
(eq (number-engines-of :a) *count*)
```

7 Interface To A Speech Recognizer

In spoken language applications, Delphi is interfaced to the output of the Byblos speech recognition system (Bates et al, 1993). The N-best paradigm is used, in which the recognizer outputs in order its top N guesses at the transcription of the sentence, for some value of N (usually 5). Delphi then runs over these transcriptions in the order they have been ranked, first with the Semantic Linker disabled so that only grammatical utterances are allowed, and if none is found, runs over them again with the Semantic Linker enabled.

8 Results Of Formal Evaluation On ATIS

Our complete system including the Semantic Linker was evaluated in the December 1993 ARPA ATIS evaluation. Prior to evaluation, ATIS versions of the system's domain model, lexicon and realization rules had been developed using several thousand utterances of training data collected from users of ATIS. An approximately 1000-utterance set was held aside as a blind test set on which all participating sites were evaluated.

Error rate in this evaluation was defined as F+NA, where F was the percentage of queries answered incorrectly, and NA the percentage of queries not answered at all. There were two evaluations on the same corpus using this metric: one of NL text understanding alone, and the other of a complete spoken language system (SLS) comprised of Delphi and the Byblos recognizer. Our system achieved an official result of 14.7% on the NL test, which was the third-lowest error rate achieved. The SLS error rate was 17.5%.

Our own experiments show that using the Semantic Linker reduced our system's error rate on the NL test by 43%. This was largely achieved by dramatically lowering the no-answer rate NA from 18.7% to 2.3%. Just over 80% of this increment of sentences answered were answered correctly, so the Linker showed considerable accuracy.

9 Porting Delphi to the SPLINT Domain

The SPLINT (Speech and Language Integration) domain is concerned with Air Force units and their component aircraft, weaponry and other physical attributes of aircraft, ordnance, and facilities (such as air bases, runways, bunkers, etc.). The SPLINT database has 106 fields in 23 tables.

Some example utterances in the SPLINT domain are:

What aircraft types are assigned to the 32nd?
Which base has a unit carrying mavericks?
Can a Stealth use Langley's runway 1?

In order to port Delphi to the SPLINT domain, SPLINT-specific versions of the domain model, lexicon, realization rules and db-mapping rules were needed. For the speech-understanding part of the application, word pronunciations were also neccesary, as well as word-class membership for a statistical n-gram class grammar. Delphi includes "core" versions of some of these knowledge bases: a core domain model with common classes like NUMBER and TIME-OF-DAY and relations like GREATER, a core lexicon with closed-class items such as prepositions as well as words appropriate to question-answering in general such as "show", to which domain-specific items have to be added.

In porting to SPLINT, 60 classes and 65 relations were added to the domain model. 400 words were added to the lexicon. Of these, approximately half were derived from database field values. 118 realization rules were added.

The grammar did not need to be modified, with the exception of adding one rule (for constructions such as "Mach 1").

The entire process took about a person month to get 90% coverage on a 1400 sentence corpus, developed independently by a non-NL person. An additional person week was required to develop the speech-related knowledge bases. A complete spoken language system with Delphi as the understanding component, plus a Motif-based user interface, was succesfully demonstrated at the 1994 ARPA Human Language Technology meeting, and at Rome Labs in New York. The porting process is described in more detail in (Bates, 1994).

This effort demonstrates that, given an appropriate system design, it is possible to build a complete spoken language system that is robust to speech and production errors, and to do so rapidly and straightforwardly.

10 Conclusion And Summary

In conclusion, we have developed a technology that makes maximal use of general linguistic knowledge to improve portability, while at the same time maintaining robustness in the face of the type of input one can expect from a real-life spoken language application. The system has been shown to reach high levels of performance in objective blind-test evaluation on the ATIS domain. The system has also been shown to be rapidly portable to a new domain, SPLINT. This did not require any changes in the underlying system code, and was done with a relatively small effort.

This work shows that computational linguistic methods, based on general knowledge of language, can be used in large, robust spoken language systems, and that special-purpose NL understanding systems do not have to be built for each new task.

11 Acknowledgments

The work reported here was supported by the Advanced Research Projects Agency and was monitored by the Office of Naval Research under Contract No. N00014-92-C-0035. The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of the Defense Advanced Research Projects Agency or the United States Government.

References

Ades, A. E. and Steedman, M. J. 1982. On the Order of Words. In *Linguistics and Philosophy 44.3, 1982, pp. 517–558.*

Alshawi, Hiyan (ed). 1992. *The Core Language Engine*, MIT Press, Cambridge.

Bates, Madeleine et al. 1993. The BBN/HARC Spoken Language Understanding System. In *Proceedings of IEEE ICASSP-93* Minneapolis, MN, April 1993, pp. 111-114, vol. II.

Bates, M. Beginning to Port a Spoken Language Database Interface In *4th Annual Dual Use Technologies and Applications Conference, Utica NY May 1994*

Bobrow, Robert. 1991. Statistical Agenda Parsing In *Proceedings 4th DARPA Workshop on Speech and Natural Language*

Dowding, John et al. Gemini: A Natural Language Understanding System for Spoken Language Understanding. In *Proceedings of the 31st Annual Meeting of the Association for Computational Linguistics*, Columbus, OH

Jackson, Eric, Appelt, Douglas, Bear, John, Moore, Robert, and A. Podlozny. A Template Matcher for Robust NL Interpretation. In *Proceedings Speech and Natural Language Workshop February 1991*

Pereira, F. C. N. and Warren, D. H. D. 1980. Definite Clause Grammars for Language Analysis—A Survey of the Formalism and a Comparison with Augmented Transition Networks. In *Artificial Intelligence 13, 1980, pp. 231–278.*

Schabes, Y., Abeille, A., and Joshi, A. K. Parsing Strategies with 'Lexicalized' Grammars': Application to Tree Adjoining Grammars. 1988 In *COLING Budapest: PROCEEDINGS of the 12th International Conference on Computational Linguistics*, Association for Computational Linguistics, Morristown, NJ, 1988, pp. 578–583.

Shieber, S. M. 1986. *An Introduction to Unification-Based Approaches to Grammar.* Center for the Study of Language and Information, Stanford, CA, 1986.

Stallard, David and Bobrow, Robert. 1991. The Mapping Unit Approach to Subcategorization. In *Proceedings Speech and Natural Language Workshop March 1991*

Stallard, David and Bobrow, Robert. 1992. Fragment Processing in the DELPHI System. In *Proceedings Speech and Natural Language Workshop February 1992*

Stallard, David and Bobrow, Robert. 1993. The Semantic Linker – a New Fragment Combining Method. In *Proceedings Human Language Technology Workshop March 1993*

Woods, William A., Kaplan, Robert A., and Nash-Webber, B. 1978. *The Lunar Sciences Natural Language Information System: Final Report.* Report 2378, Bolt, Beranek and Newman, Cambridge, MA.