STRATEGIES FOR EFFECTIVE PARAPHRASING

Marie Meteer
Varda Shaked

BBN Laboratories, Inc.
10 Moulton Street
Cambridge, Massachusetts 02238
USA

ABSTRACT

In this paper we present a new dimension to paraphrasing text in which characteristics of the original text motivate strategies for effective paraphrasing. Our system combines two existing robust components: the IRUS-II natural language understanding system and the SPOKESMAN generation system. We describe the architecture of the system and enhancements made to these components to facilitate paraphrasing. We particularly look at how levels of representation in these two systems are used by specialists in the paraphraser which define potential problems and paraphrasing strategies. Finally, we look at the role of paraphrasing in a cooperative dialog system. We will focus here on paraphrasing in the context of natural language interfaces and particularly on how multiple interpretations introduced by various kinds of ambiguity can be contrasted in paraphrases using both sentence structure and highlighting and formating the text itself.

1. INTRODUCTION[1]

While technically paraphrasing is simply the task of restating the meaning of a text in a different form, it is crucial to consider the purpose of the paraphrase in order to motivate particular strategies for changing the text. If the point of the paraphrase is to clarify the original text, as in a natural language (NL) interface to a database (DB) or expert system application, then disambiguating the query and choosing more precise lexical items (perhaps closer to the structure of the actual DB, expert system, or other underlying application) are essential strategies. If the point is to summarize information, then strategies for evaluating the relative importance of the information presented in the text are necessary. If the point is merely to restate the text differently than the original, perhaps merely to exercise the system, then one must use strategies which consider what structures and lexical items were actually found by the parser.

Our motivation for work on strategies for effective paraphrasing comes from the recent availability of NL interfaces as commercial products. As the underlying systems that a NL interface must interact with increase in number and sophistication, the range of NL interactions will increase as well. Paraphrasers developed in the past (e.g. McKeown's Co-op and BBN's Parlance™ NL Interface) were all limited in that each used only a single strategy for paraphrasing regardless of what problems may have been present in the original query. (We discuss these systems in detail in Section 6.) Our approach is to develop a variety of strategies which may be employed in different situations. We introduce a new dimension to paraphrasing text in which characteristics of the original text plus the overall context (including the goal of the system) motivate strategies for effective paraphrasing.

Our focus here will be on paraphrasing ambiguous queries in an interactive dialog system, where contrasting multiple interpretations is essential. In order to ground our discussion, we first look briefly at a range of ambiguity types. We then provide an overview of the architecture and description of the two major components: the IRUS-II™ understanding system and the Spokesman generation system. We look closely at the aspects of these systems that we augmented for the paraphrasing task and provide a detailed example of how the system appreciates multiple interpretations and uses that information to govern decision making in generation. Next we discuss the role of paraphrasing in a cooperative dialog system, and in the final section we contrast our approach with other work in paraphrasing.

[1] We would like to thank Lance Ramshaw for his invaluable help in understanding the inner workings of RUS and suggestions of where it could be augmented for our purposes, and Dawn MacLaughlin for her implementation of Parrot, the initial version of our paraphraser. We would also like to thank Ralph Weischedel, Damaris Ayuso, and David McDonald for their helpful comments of drafts of this paper and Lyn Bates for early inspirations.

2. PROBLEMS AND STRATEGIES

Ambiguity is one of the more difficult problems to detect and correct. In this section we look at three kinds of ambiguity: lexical, structural and contextual, and discuss potential strategies a paraphraser might use to eliminate the ambiguity.

1) LEXICAL AMBIGUITIES are introduced when a lexical item can refer to more than one thing. In the following example "Manhattan" can refer to either the borough of New York City or the ship:

What is the latitude and longitude of Manhattan?

The paraphraser must appreciate the ambiguity of that noun phrase, decide how to disambiguate it, and decide how much of the context to include in the paraphrase. One strategy would be to repeat the entire query, disambiguating the noun phrase by using the type and name of the object:

Do you mean what is the latitude and longitude of the city Manhattan

or what is the latitude and longitude of the ship Manhattan?

However, if the query is long, the result could be quite cumbersome. A different strategy, highlighting and formatting the text to contrast the differences, can serve to direct the user's attention to the part that is ambiguous:

Do you mean list the latitude and longitude of ***the city Manhattan***

or ***the ship Manhattan****?*

2) STRUCTURAL AMBIGUITIES are caused when there are multiple parses for a sentence. Conjunction is a typical source of structural ambiguity. Modifiers of conjoined NPs may distribute over each NP or modify only the closest NP. Consider, for example, the following query:

Display the forested hills and rivers.

This query has only one interpretation in which the premodifier "forested" modifies only the noun "hills". In contrast, the following query has two interpretations:

Display the C1 carriers and frigates

In one interpretation, the premodifier "C1" may apply only to the noun "carrier"; in the other, "C1" applies to both "carriers" and "frigates". Each interpretation requires a different paraphrase strategy. In the case where the premodifier distributes, the ambiguity may be eliminated by repeating the modifier: *Display the C1 carriers and C1 frigates*. When it does not distribute, there are three potential strategies:

--changing the order of the conjuncts: *Display the frigates and C1 carriers.*

--introducing explicit quantifiers: *Display the C1 carriers and all the frigates.*

--moving premodifiers to postmodifiers: *Display the carriers which are C1 and the frigates.*

3) CONTEXTUAL AMBIGUITIES are introduced when the query is underspecified for the underlying system it is working with. For example if the context includes a map and the possibility of natural language or table output, the query *Which carriers are C1?* could mean either *list* or *display*.

This work was supported by the Strategic Computing Program, DARPA contract number N000014-85-C-00016.

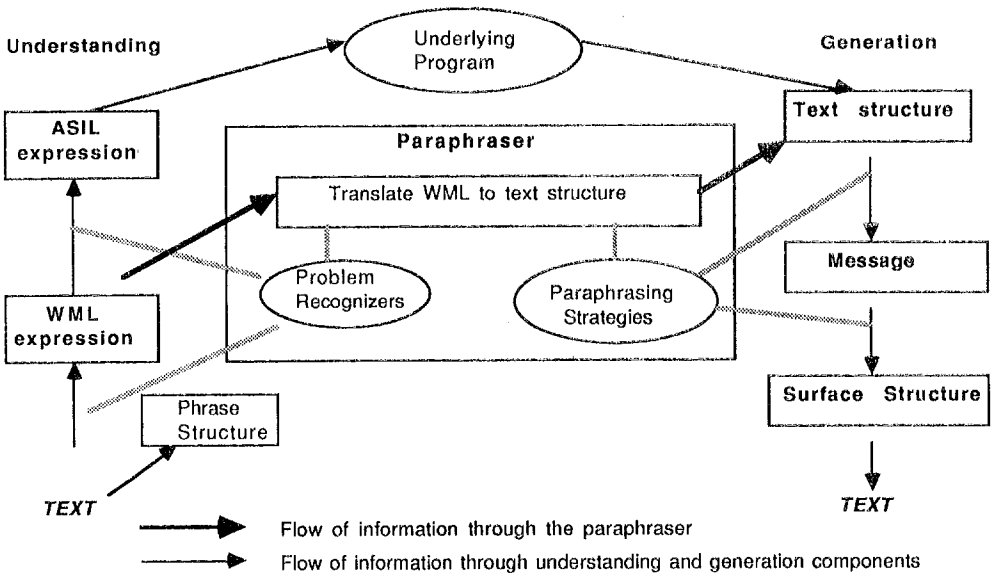


FIGURE 1 ARCHITECTURE OF THE PARAPHRASER

3. ARCHITECTURE

As the examples above illustrate, the information needed to notice problems such as ambiguity in a query is quite varied, and the strategies needed to generate a motivated paraphrase must be employed at various levels in the generation process. A distinguishing feature of our system is that it works in cooperation with existing understanding and generation components and allows the paraphraser access to multiple levels of their processing. This multilevel design allows the understanding system to appreciate ambiguities and vagueness at lexical, structural, and contextual levels, and the generation system to affect the text's organization, syntactic structure, lexical items and even to format and highlight the final text.

Figure 1 shows an overview of the architecture of the system. In this section, we first describe the understanding and generation systems independently, focusing on how the Problem Recognizers and Paraphrasing Strategies have been incorporated into the components. We then look at the paraphraser itself and how it evolved.

3.1 THE UNDERSTANDING COMPONENT: IRUS-II(TM)

IRUS-IItm (Weischedel, et al. 1987) is a robust NL understanding system that interfaces to a variety of underlying systems, such as DB management systems, expert systems and other application programs. It is capable of handling a very wide range of English constructions including ill-formed ones.

3.1.1 IRUS-II - Components and Design Principals

IRUS-II has two major processing levels which distinguish the linguistic processing from the details of the particular underlying systems it is used with. The first level, the "Front End", integrates syntactic and semantic processing. The major domain-independent "Front End" modules include a parser and associated grammar of English, a semantic interpreter, and a subsystem for resolving anaphora and ellipsis. These modules simultaneously parse an English text into a syntactic structural description and construct a formal semantic representation of its meaning in a higher order intensional logic language called the World Model Language (WML). The syntactic processor is the RUS Parser/Grammar which is based on the ATN formalism. Constants in the WML are concepts and predicates from a hierarchical domain model represented in NIKL (Moser 1983).

The more domain-dependent modules of the Front End are the lexicon, domain model, and a set of semantic Interpretation Rules (IRules). The lexicon contains information about parts of speech, and syntactic and morphological features needed for parsing, and word and phrase substitutes (such as abbreviations). An IRule defines, for a word or (semantic) class of words, the semantically acceptable English phrases that can occur having that word as a head of the phrase, and in addition defines the semantic interpretation of an accepted phrase. Thus, when the parser proposes (i.e., TRANSMITs) an intermediate syntactic phrase structure, the semantic interpreter uses the IRules that are associated with the head of that phrase to determine whether the proposed structure is interpretable and to specify its interpretation. Since semantic processing is integrated with syntactic processing, the IRules serve to block a semantically anomalous phrase as soon as it is proposed by the parser. The semantic representation of a phrase is constructed only when the phrase is believed complete.

The task of the "Back End" component of IRUS-II is to take a WML expression and compute the correct command or set of commands to one or more underlying systemsin order to obtain the result requested by the user. This problem is decomposed into the following steps:

* The WML expression is simplified and then gradually translated into the Application System Interface Langauge (ASIL).
* The particular underlying system or systems that need to be accessed are identified.
* The ASIL is transformed into underlying system(s) code to execute the query.

While the constants in WML and ASIL are domain-dependent, the constants in ASIL-to-code translation system(s) code are both domain dependent and underlying-system dependent.

3.1.2 Ambiguity Handling by the IRUS-II System - Overview

In this section, we briefly describe how various kinds of ambiguities are currently handled in IRUS-II. There are at least the following kinds of ambiguities that may occur in natural language: Semantic ambiguity (lexical, phrasal, referring expressions), structural ambiguity, quantifier scope ambiguity and collective reading ambiguity. In cases of semantic ambiguity, multiple WMLs are generated from the same syntactic parse path. For example, when a word (e.g., "Manhattan") belongs to more than one semantic class in the domain model (e.g, CITY, VESSEL), two WMLs are generated from the same syntactic parse path, each referring to a different semantic class. Similarly, premodified nouns (e.g., "Hawaii ships") generate multiple WMLs, each created as a result of multiple IRules assigning several interpretations to the relation between the elements (e.g., "Ships whose home port is Hawaii", "Ships whose destination is Hawaii", or "Ships whose current location is Hawaii").

Structural ambiguities are caused by multiple syntactic interpretations and result in alternative parse paths in the RUS parser/grammar. IRUS-II identifies these ambiguities by sequentially attempting to parse the text, with each attempt following a different parse path. Note in these cases each syntactic parse path may also have multiple semantic interpretations.

3.1.3 Enhancements to IRUS-II for Effective Paraphrasing

Though IRUS-II produces multiple interpretations (WMLs) for a variety of ambiguous sentences, it was not originally designed with the intent of paraphrasing those interpretations. While each individual WML could be paraphrased separately, a more useful approach would be to combine closely related interpretations into a single paraphrase that highlights the contrasts between the interpretations. The need to keep associations between multiple interpretations motivated the following enhancements to the IRUS-II system:

* Predefined ambiguity specialists that detect and annotate potential problems presented by the input text are "distributed" in the parser/grammar and the semantic interpreter. For example, when the parser TRANSMITs the phrase "Manhattan" to the semantic interpreter as a head of a NP, two semantic classes, CITY and VESSEL, will be associated with that NP. At this point, the Lexical Ambiguity Specialist records the lexical item "Manhattan" as the ambiguity source and the two different classes.

* After recording the potential ambiguity source, each ambiguity specialist monitors a predefined sequence of TRANSMITs associated with that source, and records the different intermediate WML expressions resulting from these TRANSMITs. For example, the Lexical Ambiguity Specialist monitors the TRANSMITs of "Manhatten" as a head noun of the NP. In this case, there will be two applicable IRules, one defining "Manhattan" as a CITY and the other defining "Manhattan" as a VESSEL. Both interpretations are semantically acceptable, resulting in two intermediate WMLs, which are then recorded by the specialist. Upon completion of the input text, two WMLs will be created and this record will be used to annotate them with their respective differences that resulted from a common ambiguity source.

We look at the details of the specialists on one particular example in Section 4.

3.2 The Generation system: SPOKESMAN

The Spokesman generation system also has two major components: a text planner and a linguistic realization component, MUMBLE-86 (Meteer et al. 1987). Both components are built within the framework of "multilevel, description directed control" (McDonald 1983). In this framework, decisions are organized into levels according to the kind of reference knowledge brought to bear (e.g. event or argument structure, syntactic structure, morphology). At each level, a representation of the utterance is constructed which both captures the decisions made so far and constrains the future decision making. The representation at each level also serves as the control for the mapping to the next level of representation.

The text planner must establish what information the utterance is to include and what wording and organization it must have in order to insure that the information is understood with the intended perspectives. The intermediate level of representation in this component is the *text structure* , which is a tree-like representation of the organization of discourse level constituents. The structure is populated with model level objects (i.e. from the applications program) and "discourse objects" (compositional objects created for the particular utterance) and the relations between these objects. The text structure is extended incrementally in two ways:

1) expanding nodes whose contents are composite objects by using predefined templates associated with the object types (such as expanding an "event" object by making its arguments submodes);

2) adding units into the structure at new nodes. The units may be selected from an already positioned composite unit or they may be individuals handed to the orchestrator by an independently driven selection process.

Once the text structure is complete, it is traversed depth first beginning with the root node. At each node, the mapping process chooses the linguistic resource (lexical item, syntactic relation such as restrictive modifier, etc.) that is to realize the object which is the content of that node. Templates associated with these objects define the set of possibilities and provide procedures for building its portion of the next level of representation, the "message level", which is the input specification for the linguistic realization component, MUMBLE-86.

The input specification to MUMBLE-86 specifies what is to be said and constrains how it is to be said. MUMBLE-86 handles the realization of the elements in the input specification (e.g. choosing between *the ships are assigned, which are assigned,* or *assigned* depending on whether the linguistic context requires a full clause, postmodifier, or premodifier), the positioning of elements in the text (e.g. choosing where to place an adverbial phrase), and the necessary morphological operations (e.g. subject-verb agreement).

In order to make these decisions, MUMBLE-86 maintains an explicit representation of the linguistic context in the form of an annotated surface structure. Labels on positions provide both syntactic constraints for choosing the appropriate phrase and a definition of which links may be broken to add more structure. This structure is traversed depth first as it is built, guiding the further realization of embedded elements and the attachment of new elements. When a word is reached by the traversal process, it is sent to the morphology process, which uses the lingusitic context to execute the appropriate morphological operations. Then the word is passed to the word stream to be output and the traversal process continues through the surface structure.

3.3 Parrot and Polly

Our first implementation of the paraphraser was simply a parrot which used the output of the parser (the WML) as input to the generator. The text planner in this case consists of a set of translation functions which build text structure and populate it with composite objects built from WML subexpressions and the constants in the WML (concepts and roles from IRUS-II's hierarchical domain model). The translation to text structure uses both explicit and implicit information from the WML. The first operator in a WML represents the speech act of the utterance. For example, BRING-ABOUT indicates explicitly that the matrix clause should be a command and implicitly that it should be in the present tense and the agent is the system. The IOTA operator indicates that the reference is definite and POWER indicates it is plural.

A second set of templates map these objects to the input specification for the linguistic component, determining the choice of lexical heads, argument structures, and attachment relations (such as restrictive-modifier or clausal-adjunct).

Interestingly, PARROT turned out to be a conceptual parrot, rather than a verbatim one. For example, the phrase *the bridge on the river* is interpreted as the following WML expression. The domain model predicate CROSS represents the role between *bridge* and *river* since IRUS interprets "on" in this particular context in terms of the CROSS relation:

(IOTA JX124 BRIDGE (CROSS JX124 (IOTA JX236 RIVER)))

This is "parroted" as *the bridge which crosses the river.* While in some cases this direct translation of the WML produces an acceptable phrase, in other cases the results are less desirable. For example, named objects are represented by an expression of the form (IOTA *var type* (NAME *var name*)), which, translated directly, would produce *the river which is named Hudson.* Such phrases make the generated text unnecessarily cumbersome. Our solution in PARROT was to implement an optimization at the point when the complex object is built and placed in the text structure that uses the name as the head of the complex object rather than the type. (Melish, 1987, discusses similar optimizations in generating from plans.)

While PARROT allowed us to establish a link from text in to text out, it is clear this approach is insufficient to do more sophisticated paraphrasing. POLLY, as we call our "smart" paraphraser, takes advantage of the extra information provided by IRUS-II in order to control the decision making in generation.

One of the most common places in which the system must choose carefully which realization to use is when the input is ambiguous and the paraphrase must contrast the two meanings. For example, if a semantic ambiguity is caused by an ambiguous name,

as in *Where is Diego Garcia* (where Diego Garcia is both a submarine and a port), the type information must be included in the paraphrase:

Do you mean where is the **port** *Diego Garcia*
or the **submarine** *Diego Garcia.*

Note, with the optimization of PARROT described above, this sentence could not be disamiguated.

In order to generate this paraphrase contrasting the two interpretations, the system needs to know what part is ambiguous at two different points in the generation process: in the text planner when selecting the information to include (both the type and the name) and at the final stage when the text is being output (to change the font). Our use of explicit active representations allows the system to mark the contrast only once, at the highest level, the text structure. This constraint is then passed through the levels and can affect decisions at any of the lower levels. Thus the system makes use of the information provided by the understanding system when it is available and ensures it will still be available when needed and won't be considered in parts of the utterance where it is not relevant.

4. Paraphrasing Syntactic Ambiguities - an Example

To elucidate the description above, we will return to an earlier example of a query with an ambiguous conjunction construction: *Display all carriers and frigates in the Indian Ocean.* This sentence has two possible interpretations:

1) *Display all carriers in the Indian Ocean and all frigates in the Indian Ocean.*

2) *Display all frigates in the Indian Ocean and all the carriers.*

In this example we show (1) how the Problem Recognizers discover that there are two interpretations and what the particular differences are; and (2) how the Paraphrasing Strategies use that information in the translation to text structure and the generation of the paraphrase.

4.1 Phase 1: The Problem Recognizers

As we discussed earlier, problem recognizing specialists have been embedded in the understanding system. Here we look at the NP Conjunction Ambiguity specialist and the two parse paths that correspond to the parses resulting from a NP conjunction ambiguity (see Figure 2 below).

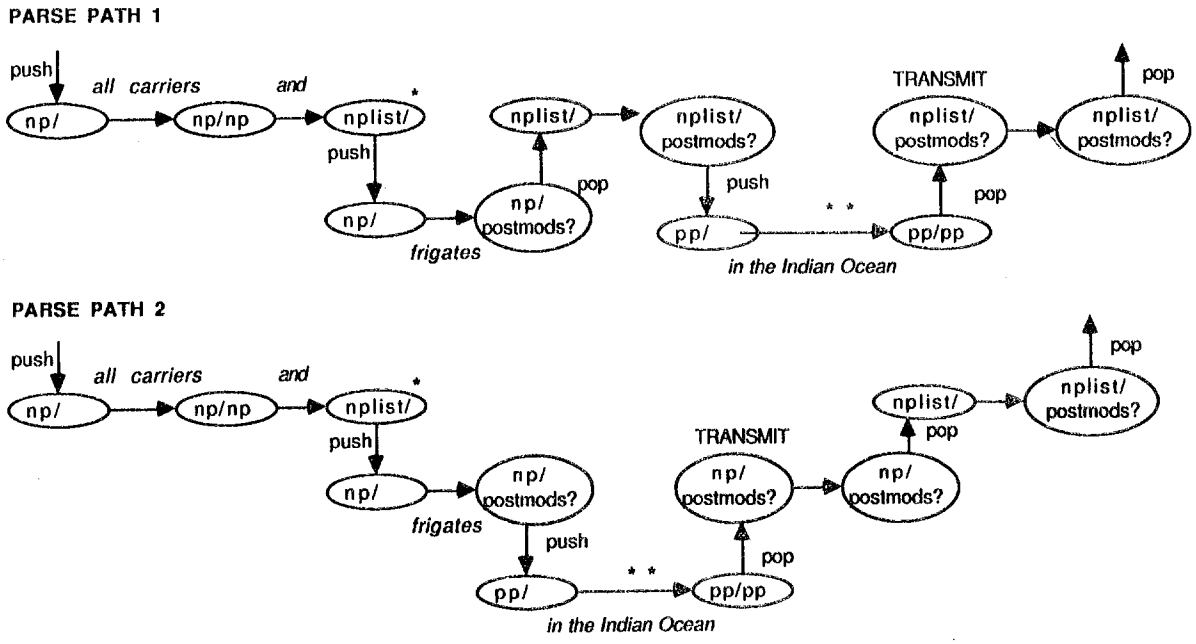
The first task of this specialist is to annotate the parse path when a NP conjunction is encountered by the parser. In IRUS-II, when the RUS parser has completed the processing of the first NP *the frigates* and the conjunction word *and*, it attempts (among other alternatives) to parse the next phrase as a NP. At this point the Conjunction Ambiguity Specialist annotates that parse path with a NP-CONJUNCTION-AMBIGUITY tag (depicted in Figure 2 with * at the first NPLIST/ state in both parse paths 1 and 2). This annotation will allow the different interpretations that may result from this NP conjunction to be grouped later according to their common ambiguity source. (Note that if we were not using an ATN, appropriate annotations could still be made using structure building rules associated with the grammar rules). The paraphraser can then organize its paraphrases according to a group of related ambiguous interpretations. As previously stated, presenting closely related interpretations simultaneously is more effective than presenting randomly generated paraphrases that correspond to arbitrary parse paths.

The second task of the NP Conjunction Ambiguity specialist is to monitor those TRANSMITs to the semantic interpreter that may result in multiple interpretations (WMLs) from the same source of ambiguity. Thus, starting from when the possible ambiguity has been noticed, this specialist will monitor the TRANSMITs to all the modifiers of the NPs. In our example, the NP Conjunction Ambiguity specialist monitors the TRANSMITs of the prepositional phrase (PP) *in the Indian Ocean* to all NPs annotated with the NP-CONJUNCTION-AMBIGUITY tag (TRANSMITs are illustrated with **), which include the TRANSMITs of that PP as a postmodifer to each of the conjoined NPs (parse path 1) as well as to only the second NP (parse path 2). Since the PP *in the Indian Ocean* is semantically acceptable as a postmodifer in both parse paths, two intermediate WMLs are be created:

Intermediate WML-1:
```
(SETOF   (IOTA ?JX19 (POWER CARRIER)
             (UNITS.LOCATION ?JX19 IO))
         (IOTA ?JX20 (POWER FRIGATE)
             (UNITS.LOCATION ?JX20 IO)))
```

Intermediate WML-2:
```
(SETOF   (IOTA ?JX19 (POWER CARRIER))
         (IOTA ?JX20 (POWER FRIGATE)
             (UNITS.LOCATION ?JX20 IO)))
```

Each intermediate WML contains a SETOF operator with two arguments that represent a pair of conjoined NPs. In Intermediate WML-1 both arguments have the UNITS.LOCATION restriction, and in Intermediate WML-2 only the second argument has that



* Set conjunction ambiguity tag
** Conjunction ambiguity specialist monitors tagged transmits to semantic interpreter

FIGURE 2 PARSE PATHS

restriction. The NP Conjunction Ambiguity specialist annotates those intermediate WMLs, and the parser proceeds to complete the processing of the input text. In our example, two final WMLs are generated, one for each of the two SETOF expressions that originated from the same NP-CONJUNCTION-AMBIGUITY source:

```
WML-1:  (BRING-ABOUT
           ((INTENSION
              (EXISTS ?JX18 LIST
                 (OBJECT.OF ?JX18
                    <Interm-WML-1>)))
            TIME WORLD))

WML-2:  (BRING-ABOUT
           ((INTENSION
              (EXISTS ?JX18 LIST
                 (OBJECT.OF ?JX18
                    <Interm-WML-2>)))
            TIME WORLD))

ANNOTATION:
     (NP-CONJUNCTION-AMBIGUITY
        (Parse-Path-1 Interps (WML-1 <Interm-WML-1>))
        (Parse-Path-2 Interps (WML-2 <Interm-WML-2>)))
```

More complex sentences that contain postmodified NP conjunction may have additional interpretations. For instance, the sentence *The carriers were destroyed by frigates and subs in the Indian Ocean* may have a third interpretation in which the PP *in the Indian Ocean* modifies the whole clause. Another more complex example is: *The carriers were destroyed by 3 frigates and subs in the Indian Ocean*, in which ambiguity specialists for NP conjunction, PP clause attachment and quantifier scoping will interact. This kind of interaction among specialists is a topic for our current research on effective paraphrasing.

4.2 Phase 2: Translating from WML to Text Structure

Once the Problem Recognizers have annotated the WML, the text planner takes over to translate the intensional logic expression into the hierarchical text structure which organizes the objects and relations specified. In this example, since the input was ambiguous and there are two WMLs, there are two possible strategies for paraphrasing which apply at this step:

(1) Paraphrase of each interpretation separately (as discussed in Section 2).

(2) Combine them into a single paraphrase using formatting and highlighting to contrast the differences:

Display the carriers in the Indian Ocean and the frigates in the Indian Ocean

or the carriers in the Indian Ocean and all the frigates.

We will focus here on the second strategy, that which combines the interpretations. The text planner will begin by translating one of the WMLS and when it reaches the subexpression that is annotated as being ambiguous, it will build a text structure object representing the disjunction of those subexpressions.

As discussed in Section 3.2, the translation to text structure uses both explicit and implicit information from the WML. In this case, the translation of the first operator, BRING-ABOUT builds a complex-event object marked as a command in the present tense and the agent is set to *you*. The domain model concept DISPLAY provides the matrix verb (see text structure in Figure 3).

When the translation reaches the SETOF expression, a COORDINATE-RELATION object is built containing both subexpressions with the relation DISJUNCTION. It is also annotated "emphasize-contrast" to guide the later decision making. As this node and its children are expanded, the annotation is passed down. When the translation reaches the individual conjuncts in the expression, it uses the annotation to decide how to expand the text structure for that object. In the case where the modifier distributes, the annotation blocks any optimization that may lead to an ambiguity, and ensures both conjuncts will be modified; in the case where it does not distribute, there are two possible strategies to eliminate the ambiguity:[2]

1) Manipulating the order of the conjuncts in the text structure:

   --If only one of the conjuncts is modified and the modifier is realizable as a premodifier, then that conjunct should be placed second.

   --If only one of the conjuncts is modified and the modifier is realizable as a postmodifier, then that conjunct should be placed first.

   In this case, the paraphrase would be: *Display the frigates in the Indian Ocean and carriers.*

2) Adding a quantifer, such as "all", to the conjunct without modification by adding an adjunct DO to the second conjunct, which would result in the paraphrase: Display all the carriers and the frigates in the Indian Ocean.

We use a combination of these strategies. Figure 3 shows the partial text stucture built for this expression[3].

[2] Note that in this task of paraphrasing queries, where it is crucial that the paraphrase be unambiguious, these are strategies the generator should apply regardless of whether the original was ambiguous or not, as ambiguity may have been introduced into a conjunction by some other strategy, such as lexical choice.

[3] Objects labeled DO in the diagram indicate discourse objects which have been created for this utterance. Objects labeled DM are objects from the domain model. The creation of discourse objects allows objects to be annotated with their roles and other information not contained in the domain model (tense, number) and introduces objects which can be referred back to anaphorically with pronouns (e.g. "they" for the DO dominating the conjuncts).

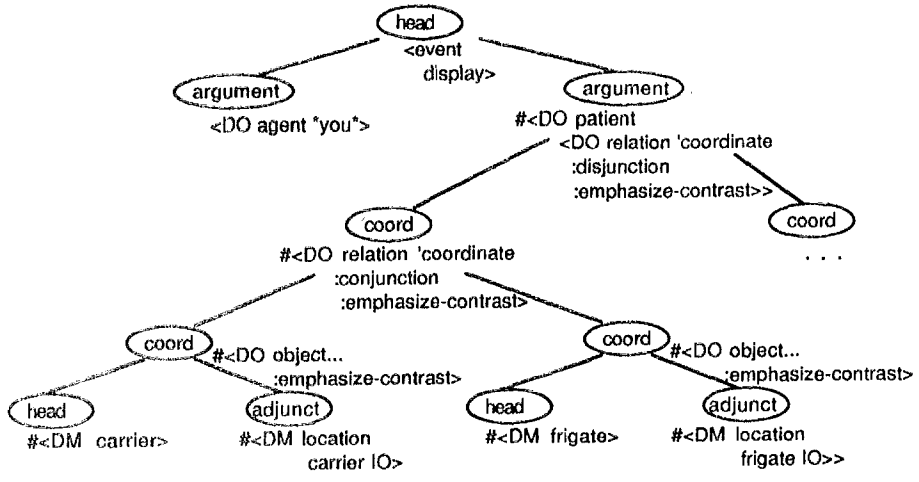


FIGURE 3: TEXT STRUCTURE FOR GENERATION

Once this level is complete, it is traversed and the linguistic resources, such as the lexical heads and major syntactic categories, are chosen and represented in the input specification to the lingusitic realization component, MUMBLE-86, which produces the final text.

5. Using the Paraphraser in a Cooperative Dialog System

The work presented here has focused on developing strategies for paraphrasing in order to resolve ambiguity. However, in an actual NL dialog system, choosing when and how to use this capability can be based on other considerations. In this section we address some practical issues and some related work we have done in the integration of our paraphraser into a Man-Machine Interface.

The presentation of a paraphrase can be useful even in cases where no ambiguity has been detected, as it allows the user to verify that the system's interpretation does not differ from the intended interpretation. This is particularly useful for new users who need to be reassured of the system's performance. This feature should be under the user's control, though, since frequent users of the system may only want to see paraphrases when the system finds multiple interpretations.

Paraphrasing can also be incorporated in cooperative responses in order to make any presuppositions explicit. Consider the following exchange:

U: Display all the carriers.
S: <icons displayed on map>
U: Which are within 500 miles of Hawaii?
S: Carriers Midway, Coral Sea, and Saratoga.
U: Which have the highest readiness ratings?
S: Of the carriers within 500 miles of Hawaii, Midway and Saratoga are C1.

Incorporating elided elements from previous queries in the response makes clear which set is being considered for the current answer.

Another sort of paraphrase, which we term "diagnostic responses", can be used when the system is unable to find any interpretation of the user's query, due to ill-formedness, novel use of language, or simply inadequate information in the underlying program. As in paraphrasing, the generator uses structures built by the understanding component to generate a focused response. For example, a metaphorical use of "commander" to refer to ships, as in the following query will violate the semantic restrictions on the arguments to the verb "assign". When IRUS-II fails to find a semantic interpretation, it saves its state, which can then be used by the generator to produce an appropriate response:

U: *Which commanders are assigned to SPA 2?*

S: *I don't understand how commanders can be assigned.*

6. Comparison with other work

A similar approach to ours is McKeown's Co-op system (McKeown, 1983). It too functions in an interactive environment. However, it is limited in several ways:

1) Since the system it worked with was limited to data base queries, it could only paraphrase questions. This is not only a limitation in functionality, but affects the linguistic competence as well: the input had to be simple WH- questions with SVO structure, no complex sentences or complicated adjuncts.

2) It had only one strategy to change the text: given and new[4], which fronted noun phrases with relative clauses or prepositional phrases that appeared in the later parts of the sentence (essentially the verb phrase). For example *Which programmers worked on oceanography projects in 1972?* would be paraphrased: *Assuming that there were oceanography projects in 1972, which programmers worked on those projects?*

3) Since its only strategy involved complex noun phrases, if there were no complex noun phrases in the query, it would be "paraphrased" exactly as the original.

[4] A related problem is that its notion of given and new was very simplistic: it is purely based on syntactic criteria of the incoming sentence and does not consider other criteria such as definiteness or context.

Lowden and de Roeck (1985) also address the problem of paraphrasing in the context of data base query. However, while they assume some parse of a query has taken place, the work focuses entirely on the generation portion of the problem. In fact, they define paraphrasing as providing a "mapping between an underlying formal representation and an NL text." They discuss in detail how text formatting can improve clarity and a solid underlying linguistic framework (in their case lexical functional grammar) can insure grammaticality. However, while they state that a paraphrase should be unambiguous, they do not address how to recognize when a query is ambiguous or how to generate an unambiguous query.

The BBN Parlance™ NL Interface is one of the most robust NL interfaces in existance. Its paraphraser integrates both the system's conceptual and procedural understanding of NL queries. This approach is based on the observation that users need to be shown the conceptual denotation of a word or phrase (e.g., "clerical employee") with its denotation in the underlying database system (e.g., an employee whose EEO category is 3 or an employee whose job title is "secretary"). Thus, the Parlance paraphrases incorporate references to specific fields and values in the underlying data base system. So, while the text can be cumbersome, it has the advantage of more directly capturing what the system understood. Due to efficiency considerations and limitations on the space for output, the Parlance paraphraser presents the paraphases one at a time, allowing the user to confirm or reject the current interpretation, rather than presenting all paraphrases at the same time. The system allows the user to refer back to previously presented interpretations, but as is the case with the other paraphrasers, related interpretations are not contrasted.

7. Conclusion

In addition to being useful in current interactive natural language interfaces, the paraphrase task provides an excellent context to explore interesting issues in both natural language understanding and generation as well as paraphrasing itself. In the next phase of our research we plan to look at quantifier scope ambiguities, lexical choice, and the interaction between multiple problems and strategies for improvement.

8. References

Hinrichs, Erhard, Damaris Ayuso, Remko Scha (1987) "The Syntax and Semantic of the JANUS Semantic Interpretation Language", Technical Report Section of BBN Report No. 6522, BBN Laboratories, pgs. 27-33.

Lowden, Barry G. T., and Anne De Roeck (1985) "Generating English Paraphrases from Relational Query Expressions", vol. 4, no. 4, p.337-348.

McKeown, Kathleen R. (1983) "Paraphrasing Questions Using Given and New Information", *American Journal of Computational Linguistics*, vol. 9. no. 1, Jan-Mar 1983, p.1-10.

McDonald, David D. (1983) "Description Directed Control", *Computers and Mathematics* 9(1), Reprinted in Grosz, et al. (eds.), *Readings in Natural Language Processing*, Morgan Kaufmann Publishers, California, 1986, p. 519-538.

Meteer, Marie M.,David D. McDonald, Scott Anderson, David Forster, Linda Gay, Alison Heuttner, Penelope Sibun (1987) *Mumble-86: Design and Implementation* University of Massachusetts Technical Report 87-87, 173 pages.

Moser, Margaret (1983) "An Overview of NIKL", Technical Report Section of BBN Report No. 5421, BBN Laboratories.

Weischedel, Ralph, Edward Walker, Damaris Ayuso, Jos de Bruin, Kimberle Koile, Lance Ramshaw, Varda Shaked (1986) "Out of the Laboratory: A case study with the IRUS natural language interface", in *Research and Development in Natural Language Understanding as part of the Strategic Computing Program*, BBN Labs Technical Report number 6463, pgs. 13-26.

Weischedel, Ralph, D. Ayuso, A. Haas, E. Hinrichs, R. Scha, V. Shaked (1987) *Research and Development in Natural Language Understanding as part of the Strategic Computing Program*, BBN Labs Technical Report number 6522.