Generation in a Natural Language Interfacef

Paul S. Jacobs

Division of Computer Science
Department of EECS
University of California
Berkeley, CA, USA

ABSTRACT

The PHRED (PHR asal English Diction) generator produces the natural language output of Berkeley's UNIX Consultant system (UC). The generator shares its knowledge base with the language analyzer PHRAN (PHRasal ANalyser). The parser and generator, together a component of UC's user interface, draw from a database of pattern-concept pairs where the basic unit of the linguistic patterns is the phrase. Both are designed to provide multilingual capabilities, to facilitate linguistic paraphrases, and to be adaptable to the individual user's vocabulary and knowledge. The generator affords extensibility, simplicity, and processing speed while performing the task of producing natural language utterances from conceptual representations using a large knowledge base. This paper describes the implementation of the phrasal generator and discusses the role of generation in a user-friendly natural language interface.

1. Some features of the system

The PHRED (PHRasal English Diction) generator is designed to produce natural language output from a conceptual knowledge structure. The generator uses a database of linguistic information shared with the language analyzer PHRAN (PHRasal ANalyser). While PHRED was originally conceived as an English generator, currently operative versions of both PHRAN and PHRED are designed for use with multiple languages.

Natural language generation is sometimes viewed as the process of planning utterances to achieve a goal (Appelt, 1982). A division in this multi-stage process can be made between the task domain and the linguistic domain, or between the system level and the interface level. PHRED operates at the interface level. User input to the UNIX consultant system (UC) is 6rst analyzed by PHRAN, producing a conceptual knowledge structure which motivates the system's response. This response, in the form of a conceptual structure, is then converted into natural language output by PHRED:

User: How do I delete a file?

UC: Typing ' rm filename ' deletes the file filename from the current directory.

User: Corao se borra un archivo?

UC: Para borrar el archhro de nombre filename del espaclo de trabajo, se escribe ' rm filename *•

User: How do I delete all the files from a directory?

UC: Typing ' rm • • will remove all files within the current directory from the current directory.

Typing ' rm -r dlrectoryname ' will cause all files and directories recursively Inside of dlrectoryname to be deleted from dlrectoryname and removes the directory dlrectoryname from the working directory.

t This research was sponsered ia part by tk« Ofkt of Naval Research aider contract NOOO14-SO-C-0732 aid tke National Science Foundation under grant IST-8007046.

The analyzer and generator form an interface module which maintains the linguistic database of the system. This module has a number of attractive features:

1.1. Dynamic capabilities

Because PHRED and PHRAN share a common database, any linguistic information added to the system may be used in both parsing and generation. New linguistic structures may be added, existing structures may be removed or amended to suit the "dialect" of the individual user, and one utterance may be explained in terms of another. This last feature is facilitated by the fact that PHRAN, while parsing, will "touch" certain patterns in the database, and PHRED can be instructed either to use or to avoid the identical patterns. The interface module has a paraphrase mode in which PHRED takes its input directly from the analyzer and rephrases:

#> John had to toll Mary that a friend kicked the bucket.

John had to inform Mary that a friend died.

#> John had to tell Mary that the bucket was kicked by a friend.

John had to inform Mary that a friend kicked a bucket.

These examples come from parts of the database not used by the UC system. UC does not yet take full advantage of PHRED's paraphrase capability; however, it can have the generator rephrase an utterance if asked for clarification by the user.

1.1. Language compatibility

The interface may draw from databases for a number of languages. The analyzer has been used with English, Spanish, and Chinese databases (Wilensky & Morgan, 1081), and both analyzer and generator currently work from English and Spanish databases. UC input and output may be in either English or Spanish, and knowledge added dynamically to the system by a user may be added in any language known by the interface.

1.3. Phrasal analysis and generation

Because linguistic knowledge in the system consists of phrasal patterns, proper handling of idioms and identification of structures similar in meaning is built into the knowledge structure. In the examples above "kicked the bucket" matches a verb pattern whose associated meaning is equivalent to that of "died"; "working directory" and "espacio de trabajo" are equivalent noun patterns.

ft. The common database

The database shared by the phrasal analyzer (PHRAN) and phrasal generator (PHRED) consists of pattern-concept pairs, where the pattern contains the linguistic structure of a phrase and the concept its internal representation. The use of the PC pair as a unit of linguistic knowledge distinguishes PHRED from some other language production mechanisms (see McDonald, 1980) in which grammatical information and conceptual information are separated. Associated with each PC pair is a list of properties used for indexing the pair in the database and for adding knowledge to the system, as well as information about forms of agreement among constituents.

Patterns are lists of constituents, where each constituent in a pattern is generally described either as a pattern of speech (p-o-s) or as a member of a descriptive class (e.g. person, physical object). Patterns may also be formed by conjunction and disjunction of other patterns and may contain specifications of constraints; for example "(and (root delete) (tense present) (voice active) (form infinitive))" is a pattern which would generate the infinitive verb "to delete". For a detailed description of PHRAN-PHRED patterns see (Wilensky & Arens, 1980).

The concepts currently used in the database are adaptations of conceptual dependency representations (see Schank, 1975) with numerous predicates, primitives, and other variations added. The concepts are not intended to represent an irreducible set of primitives or a restrictive syntax, but merely to provide a functional representation compatible with the UC knowledge base.

A simple example of a pattern-concept pair is given below:

```
(  (and (person) (with case subjective))
   (and (root remove)
        (with person (matches 1))
        (with number (matches 1)))
   (and (p-o-s noun-phrase) (with case objective))
   ([(from (container)) (object (opt-val 2))])  )
(  concept '(state-change (actor factor)
                          (state-name location)
                          (from (inside-of (object ?object)))
                          (to (not (inside-of (object ?object))))
             )
   p-o-s 'sentence
   tense (value 2 tense)
   actor (value 3)
I
```

This PC pair contains knowledge about one form of the action "remove" which can be used by the interface with the UNIX Consultant. The same pattern corresponds to both "John removed the eggs from the refrigerator" and "I removed the file foo from my directory." The pattern has four constituents. In the first, "(and (person) (with case subjective))", the constraint "(with case subjective)" indicates that if the case of the subject can be determined, it must be nominative. This is a looser constraint than "(case subjective)", which would specify that the subject *mutt* have the nominative-case property. The constraints "(with person (matches 1))" and "(with number (matches 1))" in the next constituent specify that the verb, which must have the root "remove", agrees in person and number with the subject.

ft. Implementation of the generator

The process of generation in PHRED can be divided into three phases: (1) fetching, (2) instantiation, and (3) interpretation. These three processes will be discussed in this section, with references to the "remove" pattern and to part* of the UC output from section 1. As the generation of the sample output requires complete processing of many patterns, only certain aspects of the examples can be discussed.

3.1. Fetching

While PHRAN and PHRED use the same knowledge structures, the indexing mechanisms of the analyzer and generator are naturally different. In PHRAN a pattern of speech is recognized and suggests a possible PC pair; in PHRED the same pair must be suggested by attributes of a concept or by properties of the conceptual structure.

PHRED's basic indexing mechanism uses multiple-key hashing, where primary and secondary keys are specified within the database. Keys may be arbitrary collections of properties and attributes; thus the same indexing scheme used for the most complex patterns is used for simple words. In the case of the "remove" entry above, the PC pair is indexed according to a combination of the "state-change" and the attributes "location", "inside-of, and "not inside-of". The indexing mechanism ignores variables (e.g. "Tactor"). A simpler pattern, such as the word "the", may be indexed by the properties "(p-o-s article)" and "(ref def)", indicating a definite article.

For certain groups of patterns the choice of keys is essential to the speed of fetching. In Spanish, there are numerous forms of each verb and one must be selected which agrees with its subject, sometimes with its object and indirect object as well. In other cases the choice of keys is non-essential. A default mechanism is provided where keys are not specified.

Fetching can take up most of the time used by the generator: in PHRED, running in compiled Franz Lisp on a VAX 11/780, most fetches take about 0.1 seconds of CPU time. As a typical sentence may require 20-30 fetches, most of the time used by the generator is in the fetching process.

3.2. Instantiation

The fetching mechanism returns a PC pair from a stream each time one is requested. Thus indexing on a secondary key may be delayed until an attempt is made to instantiate the PC pair(s) fetched on a primary key.

Instantiation consists of adding appropriate constraints to a selected pattern and simultaneously verifying that the pattern is appropriate for the given concept. PHRED differs from generators which use a discrimination net as an indexing tool (see Goldman, 1975) in that it combines semantic tests with the addition of constraints. This method is efficient, given that the fetching mechanism generally yields few patterns for instantiation.

The "remove" pattern in its instantiated form might be:

```
[  (and (person concept 'userl) (with case subjective))
   (and (root remove)
        (tense past)
        (with person (matches 1))
        (with number (matches 1)))
   (and (p-o-s noun-phrase concept 'fllel) (with case objective))
   ([(from (container concept 'directory1))
                                 (object (opt-val 2))))  )
```

The specification of the tokens to be represented by the individual constituents and the tense of the verb have been added to the PC pair. If these additions lead immediately to a contradiction - for example, if the token "userl" does not represent a person — the instantiation will fail.

The tense property in the second constituent has been inherited from the tense property of the entire pattern. A sentence in the past tense implies a sentence with verbs in the past tense. The process of inheritance of attributes from the overall pattern to the individual constituents may be complex: A property may be inherited by more than one constituent, as in the case of compound verb phrases, and there may be more complicated interactions between the attributes, as where a list of adjectives must be generated to describe an object.

Each fetch may return patterns which produce the desired p-o-s directly, or rewrite rules, which correspond to transformations of other patterns. The remove pattern will produce a sentence directly, but must be rewritten to produce an infinitive phrase. Generally, rewrites are tried during instantiation only after failure of the direct patterns; however, certain rewrites may be fetched immediately: If the concept indicates negation, a negating transformation will be applied to the pattern corresponding to the negated concept.

3.3. Interpretation

After a given pattern has been instantiated, the pattern can be converted into verbal output. The process of interpreting a given constituent may (1) succeed in producing a phrase as output, (2) result in a recursive application of the fetch-instantiate-interpret sequence on the given constituent , or (3) fail because of the inability of the generator to produce a specified pattern of speech. The first case occurs when the constituent represents a word or list of words for output, such as "the big apple". The second occurs if the constituent contains a more complex pattern, for example, "(and (root remove) (tense past))". The third case, where no output produces the desired pattern of speech subject to the constraints given by the uninterpreted pattern, is difficult, because the system must back up to attempt an alternate pattern without destroying too much of the work it has done. Usually this results in backing up to the level where the failed pattern was fetched, getting another pattern from the stream, and attempting instantiation of the new pattern. Most often this new pattern will be a rewrite rule, and most of the failed pattern will be used in the instantiation of the rewritten pattern. A simple case of this is where the generator fails to produce a pattern of speech for the subject of a sentence and instead generates a passive sentence.

The agreement of constituents within a pattern is assured during the interpretation phase. Constraints such as "(with person (matches 1))" are converted into properties-e.g. "(person third)". In English there are few structures where agreement passes from right to left-such as the number of a determiner which matches a succeeding noun— but in other languages there are many examples where agreement within a pattern is much more complex. In all cases PHRED can ensure proper agreement if some order of interpreting the constituents allows the correct application of constraints. In English this means that nouns within noun phrases are interpreted before their attached determiners; in more inflected languages this means that verbs must generally be produced last.

Anaphora are specially handled during interpretation . In the case of anaphoric constituents the generator applies a set of heuristics which may (1) remove the constituent entirely if it is not necessary to the utterance, (2) pronominaliie , or (3) regenerate the entire constituent. When used by *VC,* PHRED does not pronominaliie unless specifically instructed.

4. Requirements of a generator In a friendly Interface

PHRED has been designed to meet the following requirements of a generator in a natural language interface:

(1) Speak the user's language— This refers not only to the distinction between English and Spanish users, for example, but also to adaptability to the specific vocabulary of the user. A novice user will find expressions like "disk quota" and "read protection" difficult to fathom. Such phrases in the database are marked as being more technical.

(2) Clarify and/or paraphrase on demand— Where the user does not understand a particular phrase or construct, the system must be capable of providing an explanation. If the explanation can be supplied by a linguistic paraphrase, the generator provides it.

(3) Be compatible with the analyser- PHRED was designed to draw from a database of patterns used by PHRAN. The importance of this compatibility extends beyond the simple efficiency of having the same database shared by both analyxer and generator: PHRED can make use of information added dynamically by PHRAN to the database, and any new patterns constructed by the analyzer can be indexed and used by the generator.

(4) Be adaptable to the user's knowledge- The UNIX consultant's model of the user can provide PHRED with instructions which will guide word choice, proncminalization, and the degree of explicitness of the output. This is necessary to provide information comprehensible to the user and to avoid telling the user what he already knows.

5. Conclusion

PHRED represents a natural language generator rooted in a knowledge base of linguistic information shared with the PHRAN analyzer. The database, consisting of pattern-concept pairs, is maintained dynamically by the analyzer and generator. PHRED is an effective, extensible, language-independent program, and combines with PHRAN to provide a natural language interface adaptable to the individual user.

References

Appelt, D. 1982. Planning Natural Language Utterances to Satisfy Multiple Goals. SRI International: AI Center Technical Note 250.

Goldman, N. 1975. Conceptual Generation. In R. C. Schank, *Conceptual Information Pro casing.* American Elsevier Publishing Company, Inc., New York.

Hendrix, Gary G. 1977. The Lifer Manual: A Guide to Building Practical Natural Language Interfaces. SRI International: AI Center Technical Note 138.

McDonald, D. D. 1980. Language Production as a Process of Decision-making Under Constraints. Ph. D. dissertation, MIT.

Schank, R. C. 1975. *Conceptual Information Processing.* American Elsevier Publishing Company, Inc., New York.

Wilensky, R., and Arens, Y. 1980. PHRAN-A Knowledge-based Approach to Natural Language Analysis. University of California at Berkeley Electronics Research Laboratory Memorandum UCB/ERL MSO/34.

Wilensky, R. and Morgan, M. 1981. One Analyzer for Three Languages. Berkeley Electronic Research Laboratory Memorandum UCB/ERL/M81/67.