THE BBN SPOKEN LANGUAGE SYSTEM

Sean Boisen Yen-Lu Chow Andrew Haas Robert Ingria Salim Roukos David Stallard
BBN Systems and Technologies Corporation
10 Moulton Street
Mailstop 009
Cambridge, MA 02238

ABSTRACT

We describe HARC, a system for speech understanding that integrates speech recognition techniques with natural language processing. The integrated system uses statistical pattern recognition to build a lattice of potential words in the input speech. This word lattice is passed to a unification parser to derive all possible associated syntactic structures for these words. The resulting parse structures are passed to a multi-level semantics component for interpretation.

INTRODUCTION

HARC, the BBN Spoken Language System (Boisen, et al. (1989)) is a system for speech understanding that integrates speech recognition techniques with natural language processing. As our integration methodology, we use *lattice parsing*. In this architecture, an acoustic processor produces a lattice of possible words that is passed to a parser which produces all possible parses for all syntactically permissible word sequences present in the lattice. These parse trees are then passed to a semantic interpretation component, which produces the possible interpretations of these parse structures, filtering out anomalous readings where possible.

THE ARCHITECTURE OF HARC

In this section, we present a more detailed outline of the general architecture of HARC:

1. An *acoustic processor*, which uses context-dependent Hidden Markov Models (HMMs) for acoustic modelling, produces a lattice of possible words in the input speech.
2. A *chart parser* uses a unification grammar to parse the word lattice and produces the set of all possible parses for all syntactically permissible word sequences in the lattice. The resulting parses are ranked by acoustic likelihood score.
3. A *multi-level semantics* component processes the parse trees.[1] This component uses 4 translation steps to derive the meaning of each parse.
   a. The parse tree is converted to an expression of EFL (English-oriented Formal Language); at this level, each word has one EFL constant; this includes words with multiple senses.
   b. Each EFL expression is translated into one or more expressions of WML (World Model Language). Where possible, ambiguous constants from an EFL expression are disambiguated and logically equivalent EFL expressions are collapsed.
   c. Each WML expression is converted to an expression in DBL (Data Base Language), which contains one constant for each file in the data base.
   d. The value of each DBL expression is computed by evaluating the expression against the database; this value is expressed in CVL (Canonical Value Language).

For speech understanding, semantics identifies the highest scoring "meaningful" sentence. This sentence is the recognized spoken utterance and its meaning is the sytem's interpretation of the input.

TRAINING AND TEST SETS

To measure the coverage of the syntactic and semantic components and the speech understanding performance of the integrated system, we use the DARPA 1000-word Resource Management Database corpus. This corpus is divided into two sets of sentences:[2] a training corpus of 791 sentences and a test corpus of

[1]This architecture and the names of the associated language levels are from the PHLIQA1 system (Bronnenberg et al. 1980).

[2]The DARPA database has well-defined training and test sets for the speech data. However, for natural language development work, there is no such such well-defined division. For the purpose of evaluating natural language work, we defined at BBN a training corpus of 791 sentences, based on 791 patterns, and a test corpus of 200 sentences, based on an independently selected set of 200 patterns. We feel these two corpora are a reasonable interim solution for the language modelling problem in the DARPA Resource Management domain.

200 sentences. Syntax and semantics development work is done on the basis of the training corpus; the test corpus is kept hidden from the system developers, to simulate novel utterances that users of the system might make. Periodically, the test corpus is run through the system, again without the developers looking at any of the sentences. However, statistics are collected on the percentage of sentences successfully processed; this number can be compared to the percentage of the training corpus processed, to see how well the system generalizes from the training (known) to test (unknown) corpus. Subsequent sections of this paper present coverage results on both training and test sets for each component of the system.

THE ACOUSTIC PROCESSOR

Since the speech understanding search in our system is decoupled into two phases—speech acoustic scoring and language model scoring, to do this overall search most efficiently, we need to ensure that sufficient computing is performed in the first stage and enough information is saved so that optimality is preserved in the later stages of processing.

To this end, our lattice computation algorithm attempts, in principle, to compute acoustic likelihood scores for all words in the vocabulary V for all time intervals t_1 and t_2. The acoustic data is typically a sequence of analyzed and vector-quantized (VQ) input spectra sampled every 10 milliseconds (Chow, et al. 1987). We model the input speech at the phonetic level using robust context-dependent HMMs of the phoneme. The acoustic model for each word in the vocabulary is then derived from the concatenation of these phonetic HMMs. Using these acoustic models of the word, one can compute the acoustic scores for all words in the input utterance using a time-synchronous dynamic time warping (DTW) procedure with beam pruning.

An integral part of the task of the acoustic processor is to produce a word lattice that can be processed efficiently by the lattice parser. To do this, we reduce the lattice size through various lattice pruning techniques. We have used three pruning techniques, which we describe here briefly. (For full details, see Boisen, et al. (1989).)

Score Thresholding:
Word hypotheses are pruned on the basis of the *unit score*: the hypothesis' acoustic score normalized by its duration; the goal is to keep only those acoustic theories with a unit score greater than some predetermined threshold, and eliminate all others. In practice, we found it nearly impossible to find a single threshold that works for all words and have adopted a strategy that uses dual thresholds—one for short, function words and another for longer, multi-syllabic words.

Subsumption Pruning:
Subsumption pruning is designed to explicitly deal with the problem of short, function words, which are acoustically unreliable and which are often found throughout the speech signal, even within longer words.. Since it is almost always the case that short words match parts of long words, not vice versa, word theories that are found completely inside another word theory, with unit score below some factor β of the parent theory, are eliminated from the word lattice.

Forward-Backward Pruning:
Forward-backward pruning is based on the familiar forward-backward algorithm for estimating the parameters of HMMs; it requires that *all* acoustic theories must be part of a complete path through the lattice, and furthermore, must score reasonably well.

Rather then determining the optimal pruning technique and using it alone, the system uses these techniques in tandem to try to produce the optimal word lattice in terms of size and information content.

THE SYNTACTIC COMPONENT

THE GRAMMAR FORMALISM

HARC uses a grammar formalism based on annotated phrase structure rules; this formalism is called the BBN ACFG (for Annotated Context Free Grammar). While it is in the general tradition of augmented phrase structure grammars, its immediate inspiration is Definite Clause Grammars (DCGs) (Pereira & Warren (1980)). In such grammars, rules are made up of elements that are not atomic categories but, rather, are complex symbols consisting of a category label and feature specifications. Features (also called arguments) may be either constants—indicated by lists in the BBN ACFG—or variables—indicated by symbols with a leading colon. Identity of variables in the different elements of a rule is used to enforce agreement in the feature indicated by the variable. An example is (features to be discussed are underlined):

$$\begin{array}{l}
\texttt{(S (0COMP) }\underline{\texttt{:MOOD}}\texttt{ (WH-) ...)} \rightarrow \\
\texttt{(NP :NSUBCATFRAME }\underline{\texttt{:AGR}}\texttt{ }\underline{\texttt{:NPTYPE}}\texttt{ ...)} \\
\texttt{(VP }\underline{\texttt{:AGR}}\texttt{ }\underline{\texttt{:NPTYPE}}\texttt{ }\underline{\texttt{:MOOD}}\texttt{ ...)} \\
\texttt{(OPTSADJUNCT :AGR ...)}
\end{array}$$

where the variable **:AGR** enforces agreement between the VP (ultimately, its head V) and the subject NP; **:NPTYPE**, agreement between the syntactic type of the subject NP and that selected by the head V of the VP; and **:MOOD**, agreement between the mood of the S and that of the VP.

In the BBN ACFG, as in DCGs, each grammatical category has a fixed number of obligatory, positional arguments. The essential difference between our formalism and DCGs is a syntactic typing system, whereby each argument position is limited to a fixed number of values. We have found that this restriction, in conjunction with the obligatory and positional nature of arguments, to be of great assistance in developing a large grammar (currently over 800 rules). By eschewing more sophisticated mechanisms such as feature disjunction, feature negation, metarules, optional arguments, and the use of attribute-value pairs—as are found in other complex feature based grammars, such as GPSG (Gazdar, et al (1985)), LFG (Bresnan (1982)), and PATR-II (Shieber, at al. (1983))—it is relatively straightforward to have a simple syntactic checker that ensures that all grammar rules are well-formed. In a grammar as large as the BBN ACFG, having the ability to automatically make sure that all rules are well-formed is no small advantage. We have so far found no need for most of the advanced facets of other complex feature based formalisms, with the possible exception of disjunction, which will probably be added in a restricted form.

An additional difference between our work and standard DCGs is a depth-boundedness restriction, which is discussed in the next section.

THE PARSING ALGORITHM

The BBN Spoken Language System uses a parsing algorithm which is essentially that of Graham, Harrison, and Ruzzo (1980), henceforth, GHR. This algorithm, in turn, is based on the familiar Cocke-Kasami-Younger (CKY) algorithm for context-free grammars. The CKY algorithm is quite simple and powerful: it starts with the terminal elements in a sentence and builds successively larger constituents that contain those already found and constructs all possible parses of the input. However, while the CKY algorithm requires that each rule introducing non-terminal symbols—essentially the parts of speech, as opposed to the terminal symbols (lexical items and grammatical formatives)—be in Chomsky Normal Form (i.e. of the form $\mathbf{A} \rightarrow \mathbf{B} \quad \mathbf{C}$, with exactly two non-terminal symbols on the right hand side), the GHR algorithm uses several mechanisms, including tables and "dotted rules", to get around this restriction. Since the GHR algorithm, like the CKY algorithm, deals with context-free grammars, rather than context-free grammars annotated with features, the use of the required feature substitution mechanism—unification—is an extension to the GHR algorithm; see Haas (1987) for full details.

One useful result of our work on extending the GHR algorithm to handle annotated context free grammars (ACFGs) is the discovery that there is a class of ACFGs, *depth-bounded ACFGs*, for which the parsing algorithm is guaranteed to find all parses and halt (Haas (1989)). Depth-bounded ACFGs are characterized by the property that the depth of a parse tree cannot grow unboundedly large unless the length of the string also increases. In effect, such grammars do not permit rules in which a category derives only itself and no other children; such rules do not seem to be needed for the analysis of natural languages, so computational tractability is maintained without sacrificing linguistic coverage. The fact that the parsing algorithm for this class of ACFGs halts is a useful result, since parsers for complex feature based grammars cannot be guaranteed to halt, in the general case. By restricting our grammars to those that satisy depth-boundedness, we can be sure that we can parse input utterances bottom-up and find all parses without the parser going into an infinite loop.

CONSTRAINING SYNTACTIC AMBIGUITY

Since the BBN ACFG parser finds all the parses for a given input, there is a potential problem regarding the number of parses that are found for each input utterance. Our experience has been that while the average number of parses per sentence is usually quite reasonable (about 2), in cases of conjunction or ellipsis the number of parses can grow wildly. In order to obtain broad coverage without explosive ambiguity, we have experimented with a version of the parser in which rules are sorted into different levels of grammaticality. In this version of the parser, parses are ranked according to the rules utilized. Initial efforts, in which ranks were assigned to rules by hand, are encouraging. A version of the grammar which included rules such as determiner ellipsis that increased ambiguity, had an average of 18 parses per sentence and a mode of 2 parses. However, when only first order parses were considered the average was 2.86 parses and the mode was 1. The parser without the extra rules and without ranking has an average of 3.95 parses and a mode of 1.

We have also experimented with utilizing statistical methods of assigning probabilities to rules, based on the frequency of occurrence of the rules of the grammar in the training corpus. Testing the results of this automatic assignment against a corpus of 48 sentences (from the training corpus) that were manually parsed, the parse assigned the top score by the probabilistic parser was correct 77% of the time. Looking only at the top 6 parses, the correct parse was present 96% of the time. Since the success rate is 96% considering only the top 6 parses, while 50% of the sentences have 6 or more parses, this suggests that this probabalistic approach is on the right track.

SYNTACTIC COVERAGE

The current ACFG grammar contains 866 rules: of these, 424 introduce grammatical formatives (such as the articles "a", "the", prepositions, etc). The remaining rules handle the general syntactic constructions of English. Coverage on the training corpus is currently 91% and coverage of the test corpus is 81% with this grammar. The version of the grammar used by the parser that utilizes ranked rules contains 873 rules. Coverage with this version of the grammar is 94% on training and 88% on test.

THE SEMANTIC COMPONENT

As a previous section noted, the semantic processing of an input query takes place in several stages. First, the output of the parser, the parse tree, is passed to the structural semantic module. This produces an expression of the logical language EFL, which may be ambiguous. The second stage of processing accepts as input an expression of EFL and returns as output zero or more expressions of the logical language WML. The EFL translation is concerned with structural semantics—in other words, just the effect of syntactic structure on meaning. The WML translation is concerned with lexical semantics—the meaning (in a given domain) of particular words.

The third steps converts the WML expression to an expression of DBL. This translation step maps between the logical structure most natural in describing an application domain and the actual structure of database files. Finally, the answer to a database query in DBL is expressed in a formula of CVL.

THE LOGICAL LANGUAGES

Each of the logical languages just mentioned—EFL, WML, DBL and CVL—is derived from a common logical language from which each differs only in the particular constant symbols which are allowed and not in the operators (the only exception is CVL, whose operators are a *subset* of the operators of this common language).

This logical language has three main properties. First, it is *higher-order*, which means that it can quantify not only over individuals, but over sets, tuples and functions as well. Second, it is *intensional*, which means that the denotations of its expressions are assigned relative to external *indices* of world and time, and it incorporates an "intension" operator which denotes the "sense" (with respect to these indices) of an expression. Third, the languge has a rich system of *types* which are used to represent semantic selectional restrictions and so serve to delimit the set of meaningful expressions in the language.

STRUCTURAL SEMANTICS

The structural semantic component uses a set of structural semantic rules, paired one for one with the syntactic rules of the grammar. An example of these rules is given below:[3]

```
S → NP VP OPTSADJUNCT
(lambda (np vp oa)
   (oa (intension ((q np) vp))))
```

This is the top-level declarative clause rule given earlier, with its corresponding semantic rule. Note that there are three variables bound in the lambda—np, vp oa—corresponding to the three terms on the right-hand side of the syntactic rule—NP, VP, and OPTSADJUNCT. During semantic interpretation, the semantic translations of these right-hand terms are substituted in for the variables np, vp and oa to make the interpretation of the whole clause.

The effect of this rule is to construct a proposition corresponding to the application of the predicate of the clause—the VP—to the subject of the clause—the NP. This proposition is modified by the optional sentential adjunct, whose semantic translation is applied to it. Examples of sentential adjuncts are phrases such as "during April", and "in Africa", as well as adverbs and more complicated modifiers.

LEXICAL SEMANTICS

The lexical semantic component is concerned with the specific meanings of a word in a subject domain, as opposed to the manner in which these meanings are combined. These specific meanings are represented by expressions of WML which are associated by rules with the constant symbols of EFL.

A recursive-descent translation algorithm returns for each node of an EFL expression a set of WML translations. The translations for a constant expression are just those dictated by its WML translation rule. The translations for a complex expression are derived by combining, in a cartesian product fashion, the translations of the parts of the expression. At each level, the set of possible translations is filtered to remove *anomalous* translations—those which involve combinations of WML expressions with incompatible semantic types.

Sentences which are semantically ambiguous (such as "They went to the bank") will simply return multiple WML translations. Sentences which have no non-anomalous WML translation (and are therefore considered meaningless by the system) will return the empty set as the result of WML translation.

[3]Feature specifications are omitted here for conciseness.

SEMANTIC COVERAGE

Currently, the semantics is able to map 75% of the training sentences and 52% of the test sentences to a WML expression. The corresponding figures for CVL are 44% for training and 32% for test.

THE LATTICE PARSER

The basic approach we have taken for speech understanding is to extend the text parser to deal with spoken input. So instead of operating on typed input, where a single word appears unambiguously at each position, the parser must now deal with speech input, which is highly ambiguous: a set of words is possible at every position with varying acoustic likelihood scores. While the data structure which is the input to the text parser is relatively simple—a list of words—the input to the lattice parser is a lattice of all the possible words which have been found. Associated with each word in the lattice (and therefore each grammatical constituent) is a set of acoustic match scores, with each score corresponding to particular starting and ending times.

Parsing now consists of building larger grammatical constituents from smaller ones and also keeping track of the resulting acoustic scores as the new constituents span longer intervals. The parser builds these larger constituents word synchronously in a way similar to the text parser. Two parsing algorithms have been implemented for the lattice parser: the first is a relatively straightforward modification of the text parser's algorithm to deal with the input word lattice. The second is similar, but supplements the text parser's algorithm with a top-down predictive filtering component. This significantly reduces the computational complexity of the lattice parsing algorithm; see Chow & Roukos (1989) for full details.

Currently, in the integrated speech understanding system, just as in the text processing system, the semantics component is applied after all parsing is done and is not interleaved with the parsing. While there are possible disadvantages to this approach—more computation may be done since semantic knowledge is not applied until late in processing—we have chosen this method of integration as our first attempt since the integration is simple and clean.

INTEGRATED SYSTEM PERFORMANCE

In this section, we present results for HARC on the standard DARPA 1000-Word Resource Management speech database (Price, et al. (1988)), with 600 sentences (about 30 minutes) of training speech to train the acoustic models for each speaker. For these experiments, speech was sampled at 20 kHz, and 14 Mel-Frequency cepstral coeffients (MFCC), their derivatives (DMFCC), plus power (R0) and derivative of power (DR0) were computed for each 10 ms, using a 20 ms analysis window. Three separate 8-bit codebooks were created for each of the three sets of parameters using K-means vector quantization (VQ). The experiments were conducted using the multi-codebook paradigm in the HMM models, where the output of vector quantizer, which consists of a vector of 3 VQ codewords per 10 ms frame, is used as the input observation sequence to the HMM.

For the purpose of making computation tractable, we applied the lattice pruning techniques described above to a full word lattice to reduce the average lattice size from over 2000 word theories to about 60[4]. At this lattice size, the probability of having the correct word sequence in the lattice is about 98%, which places an upperbound on subsequent system performance using the language models.

Grammar	Perplexity	% Word Error	% Sentence Error
None	1000	15.45	71.3
Word Pair	60	3.9	26.0
Syntax	700	7.5	38.0
+Semantics	NA	6.9	36.4

Figure 1: Recognition Performance of HARC

Figure 1 shows the results averaged across 7 speakers, using a total of 109 utterances, under 4 grammar conditions. As shown, the grammars tested include: 1) *no* grammmar: all word sequences are possible; 2) the *word pair* grammar, containing all pairs of words occuring in the set of sentences that was used to define the database; 3) the *syntactic grammar* alone; and 4) semantic interpretation for *a posteriori* filtering on the output of lattice parsing.

Note that the performance using the *syntactic* language model is 7.5% error. At a perplexity of 700, its performance should be closer to the *no grammar* case, which has a perplexity of 1000 and an error rate of about 15%. We hypothesize that perplexity alone is not adequate to predict the quality of a language model. In order to be more precise, one needs to look at *acoustic perplexity*: a measure of how well a language model can selectively and appropriately limit acoustic confusability. A linguistically motivated language model seems to do just that—at least in this limited experiment. Also, surprisingly, using semantics gave insignificant improvement in the overall performance. One possible explanation for this is that

[4]60 word theories corresponds to about 4000 acoustic scores.

semantics gets to filter only a small number of the sentences accepted by syntax. Out of the sentences which receive semantic interpretations, syntax alone determined the correct sentence better than 60 percent of the time, leaving only about 20 sentences in which the semantics has a chance to correct the error. Unfortunately, of these errorful answers, most were semantically meaningful, although there were some exceptions. Pragmatic information may be a higher level knowledge source to constrain the possible word sequences, and therefore improve performance.

ACKNOWLEDGMENTS

The work reported here was supported by the Advanced Research Projects Agency under Contract No. N00014-C-87-0085 monitored by the Office of Naval Research. 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 of the United States Government.

References

Boisen S., Y. Chow, A. Haas, R. Ingria, S. Roucos, R. Scha, D. Stallard and M. Vilain (1989) *Integration of Speech and Natural Language: Final Report*, Report No. 6991, BBN Systems and Technologies Corporation, Cambridge, Massachusetts.

Bresnan, Joan W. (1982) *The Mental Representation of Grammatical Relations*, MIT Press, Cambridge, Massachusetts.

Bronnenberg, W.J.H.J., Harry C. Bunt, S.P. Jan Landsbergen, Remko J.H. Scha, W.J. Schoenmakers, and E.P.C. van Utteren (1980) "The Question Answering System PHLIQA1", in Leonard Bolc, ed., *Natural Language Question Answering Systems*, Hanser, Munich, pp. 217--305.

Chow, Y.L., M.O. Dunham, O.A. Kimball, M.A. Krasner, G.F. Kubala, J. Makhoul, P.J. Price, S. Roucos, and R.M. Schwartz (1987) "BYBLOS: The BBN Continuous Speech Recognition System", *IEEE Int. Conf. Acoust., Speech, Signal Processing*, Dallas, TX, April 1987, pp. 89--92, Paper No. 3.7.

Chow, Yen-Lu and Salim Roukos (1989) "Speech Understanding Using a Unification Grammar", in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, Glasgow, Scotland.

Gazdar, Gerald, Ewan Klein, Geoffrey Pullum, Ivan Sag (1985) *Generalized Phrase Structure Grammar*, Harvard University Press, Cambridge, Massachusetts.

Graham, Susan L., Michael A. Harrison, and Walter L. Ruzzo (1980) "An Improved Context-free Recognizer", *ACM Transactions on Programming Languages and Systems*, 2.3, pp. 415--461.

Haas, Andrew (1987) "Parallel Parsing for Unification Grammar", *Proceedings of the Tenth International Joint Conference on Artificial Intelligence*, pp. 615--618.

Haas, Andrew (1989) "A Generalization of the Offline Parsable Grammars", *27th Annual Meeting of the Association for Computational Linguistics: Proceedings of the Conference*, Association for Computational Linguistics, Morristown, NJ.

Pereira, Fernando C. N. and David H. D. Warren (1980) "Definite Clause Grammars for Language Analysis—A Survey of the Formalism and a Comparison with Augmented Transition Networks", *Artificial Intelligence* 13, pp. 231--278.

Price, P., W.M. Fisher, J. Bernstein and D.S. Pallett (1988) "The DARPA 1000-Word Resource Management Database for Continuous Speech Recognition", *IEEE Int. Conf. Acoust., Speech, Signal Processing*, New York, NY, April 1988, pp. 651--654.

Shieber, Stuart, Hans Uszkoreit, Fernando Pereira, Jane Robinson, and Mabry Tyson (1983) "The Formalism and Implementation of PATR-II" in Grosz, Barbra J. and Mark E. Stickel (1983) *Research on Interactive Acquisition and Use of Knowledge: Final Report SRI Project 1894*, SRI International, Menlo Park, California., pp. 39--79.