STUDIA PHONOLOGICA XXVII (1993)

Development of Spoken Language Understanding System —Comparison of Syntax-driven and Keyword-driven Approach—

Tatsuya KAWAHARA, Masahiro ARAKI and Shuji DOSHITA

ABSTRACT

We are developing a spoken dialogue system that accepts speaker-independent continuous utterances and responds to them. Two approaches are adopted and compared. Syntax-driven approach first applies syntactic analysis to constrain the input and passes syntactically accepted sentence candidates to semantic analysis. Keyword-driven approach performs keyword spotting and generates a lattice of keyword candidates as the input to semantic analyzer. For syntactic analysis, we realize LR parsing based on A* search, using word-pair constraint as heuristics. For keyword spotting, heuristic language model is incorporated to constrain the input containing the keywords. Semantic analyzer is based on a semantic network and a semantic representation is constructed by traversing the paths of the network nodes that are activated as the corresponding words are recognized. Experimental results shows that syntax-driven approach is superior to keyword-driven approach that is expected to be robust.

1 INTRODUCTION

As an intelligent and user-friendly human-machine interface, we are designing and implementing a spoken dialogue system. Its platform is a personal desk-top workstation. It features a new interactive mode to the computer, using a microphone and a speaker. The system understands user utterances and makes responses to them appropriately. By continuing a dialogue, it realizes the user's demand.

In this paper, we first present two approaches we adopt for spoken language understanding: syntax-driven approach and keyword-driven approach. Then, the components to realize the system such as syntactic analyzer and keyword spotter are explained. Finally, the current experimental evaluations are shown, comparing the two approaches.

Tatsuya KAWAHARA (河原達也), Masahiro ARAKI (荒木雅弘): Research Associates, Department of Information Science, Faculty of Engineering, Kyoto University
Shuji DOSHITA (堂下修司): Professor, Department of Information Science, Faculty of Engineering, Kyoto University

2 OUTLINE AND APPROACH

2.1 System Design

We have set a task domain of personal schedule management. Its concept is a "computer secretary", which checks the user's schedule and makes liaison with others via computer networks. Present design is just a schdule database management, which includes registration, modification and query of schedule data. It is regarded as a kind of intelligent database access with spoken language. So the goal of the system is to interpret the user's demand through a dialogue into a command of the database interface.

This system is to accept continuous utterances by any speakers. However, we now impose several constraints. At one dialogue session, just one demand which corresponds to a database access command can be realized, though it may branch to another sub-demand on its way. One utterance must be composed of single sentence, though it may be broken to some extent.

The spoken dialogue system is an integration of speech processing and natural language processing as well as knowledge processing. Figure 1 shows its flowchart. The uppermost part is a user, who makes utterances to and gets responses from the system.

The left part surrounded by a broken line corresponds to speech understanding process. It inputs user utterances and extracts their meanings, considering the context

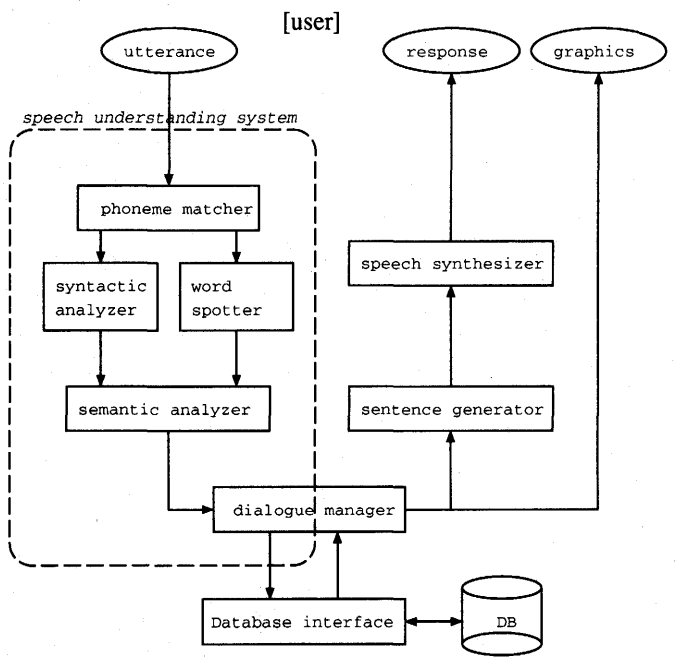

Fig. 1. Flowchart of dialogue system

of the dialogue. If it succeeds to generate a database access command, it calls the database interface program.

The right part is responding process. Dialogue manager controls generation of responding utterances, which are output to the user through speech synthesizer. Responses to the user are also done through graphic user interface if appropriate.

2.2 Flow of Understanding Process

In order to understand user utterances, we have adopted two different approaches. The two paths between phoneme matcher and semantic analyzer in Figure 1 corresponds to these approaches respectively.

Syntax-driven Approach

This approach performs syntactic analysis and obtains N-best sentence candidates, which by turns are passed to semantic analyzer to construct a semantic representation. It requires a user to utter strictly grammataical sentences, which is strong constraint but expected to improve recognition accuracy. The function of each process is explained in order.

1. Phoneme matcher

   Phoneme matcher is the front-end processor that inputs user utterances and performs phoneme recognition. However, it does not work alone, as phoneme recognition without any linguistic constraints is almost impossible. It is driven by syntactic analyzer. Namely, the parser predicts some word, which is expanded to a phonetic sequence, and asks the phoneme matcher to evaluate it. So the output of the matcher is a score of the predicted phoneme sequences.

2. Syntactic analyzer

   Syntactic analyzer is the core in syntax-driven approach. It constructs sentence hypotheses for an input utterance. It predicts possible partial sentences, calls phoneme matcher, and extends the plausible ones based on the matching results, until a complete sentence is obtained. By continuing this process, finally, N-best sentences are output to the semantic analyzer.

3. Semantic analyzer

   Semantic analyzer constructs semantic representations for given user utterances. In syntax-driven approach, it parses syntactically complete sentences by turns from the best candidate, until a correct semantic representation is obtained. Therefore, conventional parsers for written language can be used with minor modifications.

4. Dialogue manager

   Dialogue manager plays important roles in understanding utterances. It fills and even modifies some slots in semantic representations according to the

context of the dialogue. It will also provide predictive information to semantic analayzer and syntactic analyzer.

Keyword-driven Approach

This approach performs word spotting and obtains a word lattice, a set of word candidates with scores and time-alignment labels, which is passed to semantic analyzer. In this case, the semantic analyzer must parse word lattices, not complete sentences. This approach allows more flexible utterances that do not have to follow grammars. It proceeds as follows.

1. Phoneme matcher
   Same as in syntax-driven approach, except that it is driven by word spotter.
2. Word spotter
   Word spotter is the front-end processor of the approach. It recognizes words with time-aligning. It calls phoneme matcher and extends the plausible word candidates. A set of word candidates whose scores exceed some threshold are output.
3. Semantic analyzer
   The role of semantic analyzer is fundamentally same as that in syntax-driven approach. In keyword-driven approach, however, it constructs sentence hypotheses from an input word lattice. It predicts possible combinations of words and evaluates their scores and the coverage of the input speech. Then plausible ones are extended, until a semantic representation covering reasonable parts of the speech is obtained.
4. Dialogue manager
   Same as in syntax-driven approach.

3 COMPONENTS

3.1 Phoneme Matcher based on Discriminative HMM

Speech analysis with digital signal processing is performed in order to extract feataures that is speaker-independent and distinctive of phonemes. In our system, a speech sample is sampled at 16 kHz and quantized to linear 16 bit. The speech waveform is then segmented into frames. The frame length is 25 ms. The frame shift or the interval of analysis is 10 ms. Hamming window is applied to each frame. For each windowed frame, LPC analysis of 26th order is performed. We compute from the coefficients a smoothed power spectrum of 28 channels that is nearly mel-scaled. Beside this, we compute a power parameter with the residue of LPC analysis.

An input speech pattern, through the analysis, is matched with phonetic models.

A new architecture of HMM with high discriminating ability has been developed [1], [2]. In modifying continuous HMM so that states of the models are best separated, distinctive features vary for different states or different models, and different discriminant functions should be made for different competing states. In our approach, for every pair of the states, a Bayes classifier which performs a vector transformation based on discriminant analysis is constructed. Each classifier ranks the two states and computes a relative value of the probabilities. Output probabilities of the HMM states are obtained by combining and normalizing the results of the pair-wise classifications. It realizes robust recognition by modifying pattern space to fully separate confusing classes, while retaining analog outputs by statistical Bayes classifiers.

3.2 Syntactic Analyzer based on A* Search

We have realized an A* search algorithm for continuous speech recognition with phoneme HMM and context-free (LR) parser [3], [4].

The heuristic search proceeds by evaluating each hypothesis n with the following function $\hat{f}(n)$, that is a combination of the matching score of the extended part $g(n)$ and the estimate of the unsearched part $\hat{h}(n)$ called heuristics. It is illustrated in Figure 2.

$$\hat{f}(n) = g(n) + \hat{h}(n)$$

In order to satisfy A*-admissibility, or guarantee the search to find the optimal solution, the estimate must not be less than the actual score (in negative values), namely $\hat{h}(n) \geq h(n)$. However, if the estimate is far larger than the actual score, the heuristic function is not useful and the search extends unsurmountable number of hypotheses. Thus, the estimate should be as accurate as possible, namely $\hat{h}(n) \simeq h(n)$. Moreover, the estimation should be computationally cheap, because the overall efficiency is evaluated by the search efficiency itself plus the heuristics computation.

Considering the above requirements, we use a word-pair constraint as baseline heuristics. It is represented by a single automaton called word-pair HMM, a combination of the word models, where syntactically admissible pairs of the words are connected. The word-pair is derived so that it accepts a superset of the language generated by the original context-free grammar. This guarantees the A*-admissibility condition. Moreover, the word-pair constraint provides powerful heuristics whose perplexity is close to that of the original context-free grammar.

To realize this heuristic search, we adopt two pass search strategy [5]. The first pass applies computationally cheaper constraint to get heuristics, which is utilized to guide the search of the second pass, where all the available knowledge sources are applied. For heuristics computation, the word-pair model is applied from the end of the input speech to the beginning, and a backward trellis is generated. The heuristic score from any word at any time-frame to the end of the speech is computed and stored. The scores are common to all the sentence hypotheses. In the forward search, we perform LR parsing, to predict the following words of the current sentence

hypothesis and to check if the hypothesis is accepted as a complete sentence at the end. A forward trellis is generated for each sentence hypothesis by concatenating HMMs according to the transcription. The hypothesis is evaluated by the Viterbi score on concatenation of the forward trellis and the backward trellis. The search proceeds best-first, namely extends the currently most promising hypothesis. The first hypothesis that is accepted by the grammar and reaches the end of the input is guaranteed to be the optimal solution.

The search algorithm is described below.

1. A backward trellis is generated to compute the heuristic scores by the word-pair model.
2. The words that can appear at the beginning of sentences are picked up. For each of them, a new sentence hypothesis is generated and pushed to the stack with its evaluation score $\hat{f}(n)$.
3. The hypothesis n with the largest score is popped from the stack.
4. If the hypotyhesis n reaches the end of the input speech and is accepted by the given grammar, then it is the optimal solution. Finish the search.
5. The words that can follow the hypothesis n are predicted. For each of them, a new sentence hypothesis replacing n is generated and pushed to the stack with its evaluation score $\hat{f}(n)$. Go to Step 3.

This is easily extended to an N-best algorithm. If we continue the search after finding the optimal solution, we can obtain the N-best sentence candidates.

Moreover, we realize a probabilistic version of this. Namely we derive a probabilistic word-pair grammar which is subset of a probabilistic context free grammar and thus provides A*-admissible heuristics.

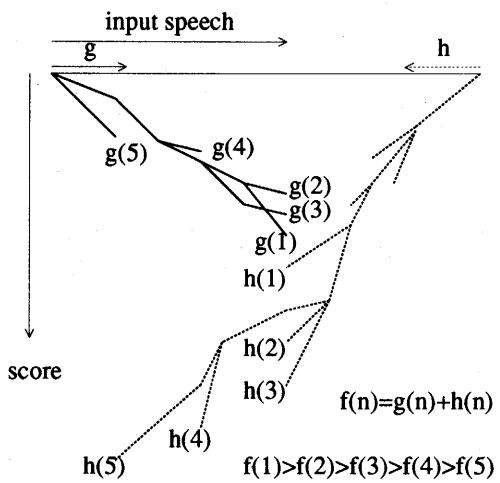

Fig. 2. Evaluation of sentence hypotheses

3.3 Word Spotter with Heuristic Language Model

The goal of word spotter for speech understanding is to spot all the signifacant words that contributes to sentence meaning in a given task domain. In this kind of multiple word spotting in a sentence utterance, we can assume that an input contains some word to be spotted as well as other words. Namely, an input is a sequence of spotted words, other (unknown) words, filled and silent pauses. Therefore, it is desirable to model the language and incorporate this knowledge into spotting phase.

In the conventional study on word spotting, a word model is matched assuming that every time-frame be starting point or end point of the word. But it is difficult to compare Viterbi scores that are different in length (scoring problem). It is also hard to precisely identify starting points and end points that give the maximum score (segmentation problem), without identifying neighboring parts. This sort of bottom-up matching is always annoyed with local noise or similarity.

We have proposed to put linguistic constraint on the whole input and to find the best hypothesis that contains the spotted word and satisfies the constraint. The evaluation function for a hypothesis that the input contains word w in time $t_1 \sim t_2$ is defined as the sum of the score for the word itself $g(w,t_1,t_2)$ and the score $h(w,t_1,t_2)$ for the rest part $1 \sim t_1$, $t_2 \sim T$. The spotting model is illustrated in Figure 3.

$$f(w,t_1,t_2) = g(w,t_1,t_2) + h(w,t_1,t_2)$$

The heuristic score h that the rest part makes plausible sentence in a task is ignored as zero in conventional strategies. Since we incorporate heuristic knowledge on language, we call this strategy heuristic word spotting. The heuristic score is divided into the preceding part $(1 \sim t_1)$ and the following part $(t_2 \sim T)$ of the word.

$$h(w,t_1,t_2) = h_l(w,1,,t_1) + h_r(w,t_2,T)$$

We call them left-context heuristics $h_l(w,1,t_1)$ and right-context heuristics $h_r(w,t_2,T)$, respectively.

This formulation solves the problems of scoring and segmentation which arise in length-free matching, as it judges after scanning the whole input. Our strategy uses a language model that is not a strict grammar but constrains inputs to be

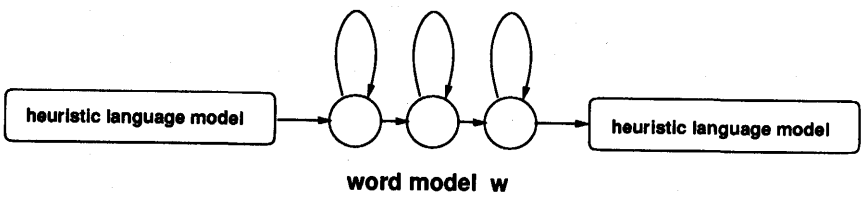

Fig. 3. Spotting model with heuristic language model

plausible sentences in a task.

Heuristic spotting algorithm consists of two phases: heuristics computation and spotting itself. First of all, the heuristic model is applied to whole part of an input speech. Although the same model is used to approximate the left part and the right part of the spotted word, Viterbi scores for the two must independently computed. Therefore, we apply the model both left-to-right and right-to-left, and stores the respective generated trellis. Thus, the overall procedure becomes a three-pass algorithm on the input including the spotting phase itself. The evaluation function $f(w,t_1,t_2)$ is obtained by concatenating the three trellises.

The key of heuristic word spotting is language modeling that can work as constraint for the spotter, but not for users. Several models are examined.

- Syllable-concatenation model

  The loosest constraint is just parallel concatenation of the possible syllables without lexical knowledge. Its model is self-looping of syllables that are possible in the language. In Japanese, there are about 100 CV (Consonant-Vowel) syllables. Therefore, this model can accept any Japanese sentences.

- Word-concatenation model

  This model approximates an input utterance with a sequence of known words including spotted words, using large vocabulary lexical knowledge [6]. It is a parallel combination of all the possible word models. To cope with spontaneous utterances, models of filled pauses are also added.

- Word-pair model

  In addition to the lexical knowledge, this model constrains the connections of the words. Only the syntactically admissible pairs of the words are connected. Syntactic constraint on the word connections is often violated in Japanese conversational speech. But this constraint will improve the spotting accuracy.

3.4 Robust Semantic Analyzer with Semantic Network

To realize flexible and robust parsing, we have developed semantic analyzer based on a network representation [7]. Semantic network represents the relation of words and their concepts, quite compactly. Taking the paths of arbitrary nodes, a semantic repesentation is obtained.

The semantic network we adopt is layer structured and loop-free, actually. The lower level represents the lexical knowledge, and the terminals or leaf nodes correspond to keywords that construct the meaning of sentences. The upper level represents the semantic and the pragmatic knowledge, and the top nodes are cores of the semantic

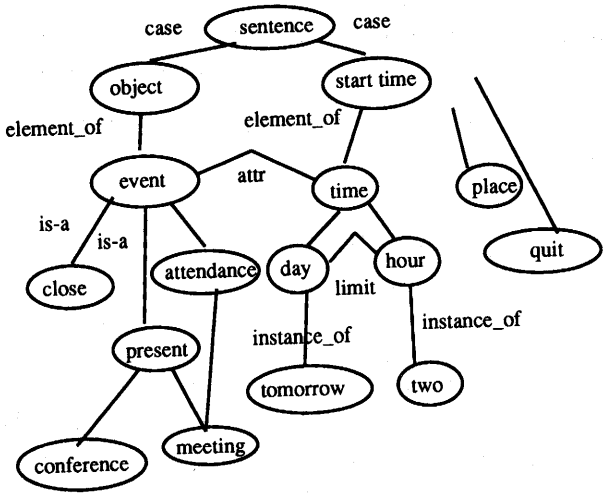

Fig. 4. A part of semantic network

types of sentences. The arcs between nodes have several types of attributes such as *is_a* and *instance_of*. A part of a network is shown in Figure 4.

The basic strategy to network parsing is classified into two types: marker passing [8] and analogue spreading activation [9]. Here we adopt the former strategy as analogue activation is difficult to formulate and estimate parameters.

When a word is recognized or spotted, the corresponding node of the network is activated. The semantic representation for the input is constructed by traversing the paths between activated nodes. An arbitrary path between two nodes makes a partial semantic representation.

For example, when we recognize "tomorrow" and "meeting", then following two paths are found in the network shown in Figure 4.

(a) tomorrow→*day*→*time*→*start_time*→*sentence*←*object*←*event*←*present*←meeting

(b) tomorrow→*day*→*time*↔*event*←*present*←meeting

They are transformed to following semantic representations, respectively.

(a) [assert, [*start_time*, [*time*, tomorrow]], [*event*, [*present*, meeting]]]

(b) [assert, [*event*, [*time*, tomorrow], [*present*, meeting]]]

If necessary, syntactic dependency is checked by referring functional words.

As following words are obtained, new paths with neighboring words are generated and semantic representations are extended. Here semantic consistency is verified and contradictory hypotheses are not extended. A sentence is completed if a semantic representation is fulfilled and whole input is parsed. Words are picked up so that they cover the whole parts.

The parsing algorithm is described below.

1. Get a word. If it completes a sentence, then finish.

2. Get another word.
3. Traverse paths with neighboring words, and extend semantic representations.
4. If a hypothesis completes a sentence, then finish. Otherwise go to step 2.

This mechanism realizes robust parsing against minor recognition errors or spontaneous utterances.

4 EXPERIMENTAL EVALUATIONS

We have implemented all the described methods to build a spoken language understanding system.

We have evaluated it with two kinds of sentence sets. A set of 50 fixed sentences are chosen and each of them is uttered by 8 male speakers. Another set of 25 sentences that contain several filled pauses thus we call spontaneous sentences are also uttered by 8 male speakers. The net vocabulary size is 232, and the number of the keywords to be spotted is 220. We have also made comparison of the proposed two approaches: syntax-driven and keyword-driven. Both use the same semantic analyzer. The difference is its input is 10-best sentence candidates or a keyword lattice.

At first, we evaluated the performance of LR parser and keyword spotter themselves. It is measured by the word accuracy. As for LR parsing, the word accuracy of the 1-best and the 10-best sentence candidates is investigated. As for keyword spotter, W-best and $3W$-best word candidates are investigated, where W is the number of the words to be spotted.

The results of syntactic parsing are shown in Table 1. Here two grammars are used. Grammar GS1 has filled pauses in its syntactic rules and vocabulary, while grammar GS0 does not. The word perplexity of GS0 and GS1 is 44.4 and 47.3, respectively. For spontaneous utterances that contain filled pauses, GS0 can not cope and gets less accuracy than GS1. For fixed utterances, GS1 whose perplexity is larger due to the registered filled pauses, got a bit worse.

Table 1. Word recognition accuracy with syntactic parser

	fixed	spontaneous
grammar without filler (GS0)	81.7 (92.2)	68.8 (81.9)
grammar with filler (GS1)	80.8 (91.6)	71.7 (85.6)

upper row: in 1-best sentences candidates
lower row: in 10-best sentence candidates

The results of 220 keyword spotting are listed in Table 2. Here we compared three heuristic language models: syllable concatenation model, word concatenation model, and word-pair model. The filled pauses are also registered in their lexicon.

Table 2. Word spotting accuracy

	fixed	spontaneous
syllable model	42.2 (65.4)	41.7 (61.9)
word model (with filler)	60.9 (83.3)	56.2 (76.4)
word-pair model (with filler)	80.9 (93.6)	69.6 (84.4)

upper row: in $1W$-best word candidates
lower row: in $3W$-best word candidates

The word preplexity of word-pair model is 51.7. With the more powerful heuristic language model, the spotting accuracy gets better. But the difference of the accuracy for fixed and spontaneous utterances is little with a loose model. It should be noticed that, when we use word-pair model as heuristics, the word recognition rate is much the same as that of syntactic parsing.

Finally, we made experiments of sentence understanding, by applying semantic analysis based on the semantic network. In syntax-driven approach, the 10-best sentence candidates are analyzed by turns until a semantic representation is obtained. In keyword-driven approach, a set of spotted keywords is directly parsed by semantic analyzer. Here, we used the lattice obtained with word-pair model, which was the best in the previous experiment. For fixed utterances, syntax-driven approach gets much better accuracy by about 20%. As the word accuracy is much the same for the both approaches, it is concluded that the word lattice is so relaxed and inadequate as an interface representation. For spontaneous utterances, the grammar without filled pauses (GS0) failed to obtain correct sentence hypotheses. With adding fillers to the grammar, the accuracy is improved. This results, conversely, proves that the syntax is not robust. The keyword-driven approach did not work better than expected

Table 3. Comparison of syntactic parser and word spotter as front-end recognizer

	fixed	spontaneous	perplexity
syntax-driven approach (grammar without filler: GS0)	65.8 (82.3)	24.0 (41.5)	44.4
syntax-driven approach (grammar with filler: GS1)	64.0 (81.5)	49.0 (69.0)	47.3
keyword-driven approach (with word-pair heuristics)	44.0 (59.5)	27.5 (41.0)	51.7

perplexity: at front-end recognizer
upper row: semantic accuracy
lower row: accuracy tolerating 1 slot error

in this case, either. Although the word accuracy is the much the same in any methods performed here, there is much difference in the sentence accuracy.

5 DISCUSSIONS

Syntax-driven approach is effective for fixed patterns of utterances. However, there are problems to be solved for spontaneous speech understanding. First, syntax-driven approach is not robust against the violation of the grammar. Second, keyword-driven approach is not sufficient, especially for a large vocabulary task, as the quality of the obtained keyword lattice is poor for the following semantic analysis.

We are exploring a new approach which uses poweful constraints as in syntax-driven approach and is robust against the out-of-grammar utterances aimed by keyword-driven approach.

REFERENCES

[1] T. Kawahara and S. Doshita. Phoneme recognition by combining discriminant analysis and HMM. In *Proc. of IEEE-ICASSP*, pages 557–560, 1991.

[2] T. Kawahara and S. Doshita. HMM based on pair-wise Bayes classifiers. In *Proc of IEEE-ICASSP*, volume 1. pages 365–368, 1992.

[3] T. Kawahara, S. Matsumoto, and S. Doshita. A*-admissible context-free parsing on HMM trellis for speech understanding. In *Proc of Pacific Rim Int'l Conf. on Artificial Intelligence*, volume 2, pages 1203–1208, 1992.

[4] T. Kawahara, M. Araki, and S. Doshita. Heuristic search integrating syntactic, semantic and dialog-level constraints. In *Proc of IEEE-ICASSP*, (to appear), 1994.

[5] S. Austin, R. Schwatz, and P. Placeway. The forward-backward search algorithm. In *Proc of IEEE-ICASSP*, pages 697–700, 1991.

[6] J.R. Rohlicek, P. Jeanrenaud, K. Ng, H. Gish, B. Musicus, and M. Siu. Phonetic training and language modeling for word spotting. In *Proc of IEEE-ICASSP*, volume 2, pages 459–462, 1993.

[7] M. Araki, T. Kawahara, T. Nishida, and S. Doshita. Keyword-driven speech parser using dialog-level knowledge. In *Proc of Pacific Rim Int'l Conf. on Artificial Intelligence*, volume 2, pages 1025–1029, 1992.

[8] E. Charniak. A neat theory of marker passing. In *Proc of AAAI*, pages 584–588, 1986.

[9] D.L. Waltz and J.B. Pollack. Massively parallel parsing: A strongly interactive model of natural language interpretation. In *Cognitive Science*, volume 9, pages 51–74, 1985.