

A PROCEDURAL REPRESENTATION OF LEXICAL ENTRIES
IN AUGMENTED TRANSITION NETWORK GRAMMAR

Yutaka Kobayashi and Yasuhisa Niimi
Dept. of Computer Science
Kyoto Technical University
Sakyo-ku Matsugasaki
Kyoto 606, JAPAN

The authors have developed a new representation method of lexical entries and implemented it in the continuous speech recognition system. The basic structure of an entry is a transition network. Associated with each arc of the network are the expected phoneme symbol and a pair of procedures which specify the run-time treatment of phonological transformations and provide the facility of arbitrary contextual and prosodic tests.

1. INTRODUCTION

We have been working on the development of a continuous speech recognition system of a subset of 'BASIC', aiming at a voice-input programming system. The major items of task specifications are as follow: continuous speech, high S/N ratio recording, strict artificial syntax, vocabulary size of 50 words, loose semantic constraints, and multiple speaker. The first version of our system [1], [2] demonstrated a sentence recognition rate of 85 percent, comparable to any other system, but not adequate in a real situation.

The main difficulties of almost all speech recognition systems reported so far lie in the precision and accuracy of the acoustic analyzer, the representation of the word dictionary and the word matching method in the context of ambiguous phoneme and word boundaries. Although the importance of higher linguistic constraints was certified as expected, the improvement of the fundamental power of the system is now again, strongly asked for. In particular, the proper use of prosodic information is expected.

To cope with it, we have improved the acoustic analyzer and developed a new lexical matching module based on a procedural representation of lexical entries in the Augmented Transition Network Grammar (ATNG). The skelton of a lexical entry is a transition network in which a transition along an arc is conditioned by the recognition of a constituent phoneme. We also attach to each arc a pair of procedures written in LISP-like format which specify the run-time treatment of phonological transformations and

provide the facility of arbitrary contextual and prosodic tests including any interaction with other levels of the system.

The comparison with relevant works is given later in Section 4.

Before entering into the details, we should better state the environment around the lexical level of our system. It receives a segment lattice from the acoustic analyzer and the possible words predicted by the parser, and returns the matched words with their score back to the parser. The parser works in the top-down, left-to-right and best-first manner.

2. REPRESENTATION OF LEXICAL ENTRIES

The ATNG [3] has been used to describe higher linguistic information such as syntax and semantics. A network is a graph made of several nodes and directed arcs connecting them. There are four types of arcs, CAT, PUSH, POP and TST, and a procedure may be associated with an arc, which operates on the global and local variables containing partial results of the analysis.

To deal with our problems we have made a few changes in notation and interpretation. An arc is a quadruple (DEST, PHON, TRUE, FALSE). PHON is a single phoneme label or a procedure which contains a precondition of the phoneme searching and a specification of one or more phonemes. The discovery of one of these phonemes causes a transition to the destination node DEST of the network. TRUE is a procedure which is carried out after the searching, especially when one of the predicted phonemes

is found in the specified region of the acoustic output. The procedure mainly consists of some tests about the matched phoneme and global features such as a syllable duration. In recognizing a short functional word, it is often the case that prosodic cues, such as the phoneme duration, play an important role. If the phoneme is not found or a test of TRUE procedure rejects the phoneme matching result as inadequate, the FALSE procedure is performed instead of the immediate truncation of that path in the word graph. Some of phonological transformations are also treated here. Some penalty increment is given to the omission of the phoneme, and the replaceable secondary phoneme is looked for if it is expected. Thus, the invention of TRUE and FALSE combination offers simple graph structures as well as appropriate interpretation of phonological transformations.

We adopted LISP-like representation to describe procedures. Atoms correspond to constants and, what we call, registers. Only numerical constants are allowed. The registers are divided in two types: local registers and global ones. The former type includes several general registers and those containing the information on the specific phonemes matched in the limited region of the acoustic output. For instance, we use a general register to hold the accumulated value of segment durations for the test of syllable duration. Registers of the latter type are chiefly those containing information on the neighboring word. As basic operations, we have comparing operators between atoms, their boolean connections, register setting and conditional execution control (COND-function). Other basic functions are penalty incrementation, phoneme prediction, additional phoneme searching and so on. Any combination of the basic functions is possible and, moreover, arbitrary processings can be added in the manner of subroutine calls. For instance, one could ask for a finer acoustic analysis of the short interval of speech if it might be critical to distinguish the predicted words.

3. LEXICAL MATCHING CONTROL

The lexical matching control is a simulator of word acceptors. The matching is straightforward in the sense that a transition from a node to its successor takes place when a phonemic symbol on the outgoing arc is found in the acoustic output. Since the number of likely phonemic symbols is limited even if the number of predicted words grows, the control

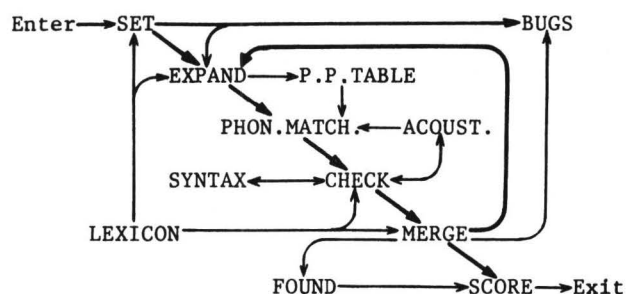


Fig.1 Flow of Lexical Matching Control

performs all the word matchings in parallel to reduce the effort. Fig. 1 illustrates the flow of the control.

To begin with, we introduce a notion of a BUG which is a quadruple (NODE, START, SEG, PEN). NODE is a pointer to a node of the lexical entry, initially set equal to the initial node. START denotes the starting segment of the word to be verified, while SEG is a pointer to the current segment of phoneme matching, initially set equal to START. PEN contains the accumulated value of word matching penalty.

SET : The control generates as many BUGs as the number of predicted words multiplied by the number of starting segments. These BUGs are placed on the BUG list.

EXPAND : Then it picks up all the BUGs having the smallest segment number SEG, and makes a phoneme prediction table which contains all the phonemes on the arcs going out from the lexical NODEs of the selected BUGs. The contextual tests may be done in advance of the phoneme prediction on some arcs.

PHONEME MATCHING : In this step, each phonemic accordance is searched against the acoustic output. Starting from SEG, the search for the specific phoneme proceeds in the segment lattice. Short transient segments are allowed to skip. The score of phoneme matching is calculated as a duration-weighted average of component label ratings, and the phoneme matching penalty is a function of the score and the skipped gap duration. In order to obtain the optimal word matching over ambiguous boundaries, we do not select a single phoneme matching result here.

CHECK : Receiving duration and scores of all

the predicted phonemes, the next step is to perform the TRUE or FALSE procedure on the relevant arc of the lexicon. The detail of this is stated in the preceding section. When passing the test, a new BUG is generated in which NODE is replaced by its successor DEST, and SEG and PEN are updated. The procedure gives an additional increment to PEN in some cases.

MERGE : Those BUGs picked up in "EXPAND" are removed from the BUG list and the newly generated ones are added to it. If several BUGs in the list have an identical NODE number of a same word and an identical SEGment number, the one having the smallest PENalty is selected among them, while the others are discarded. After the optimization is done in such a way, the BUGs which have arrived at the final node are moved to the FOUND list. Then the control returns to "EXPAND" to repeat the cycle, if the BUG list is not empty. Otherwise, it goes to the next step.

SCORE : If the FOUND list is not empty, the word matching score is calculated for each matched word from the accumulated PENalty, and the plural results of the same word having the same START segment are merged to one with a range of end segments. This greatly reduces the search space of the parser.

Thus the lexical matching is performed in such a manner that many BUGs proceed deviding, uniting and dying in the product space spanned by the segment lattice and all the predicted word graphs. In other words, this matching process is equivalent to the breadth-first search in this product space. Moreover, it is considered to be an application of the dynamic programming matching method because PEN is a non-decreasing function and the optimization is done at several points in the space.

4. EXPERIMENTAL RESULT AND DISCUSSIONS

Implementing the system in a minicomputer in Assembly language, we have carried out the recognition experiment of 72 statements of 'BASIC' language read by 2 speakers in order to estimate the facility of the new lexical representation. At first we designed very simple lexical entries from orthographic pronunciations, and then refined them observing the acoustic output of one of the two speakers. The refinement includes duration and contextual tests, creation of optional arcs and nodes, and so on. Experimental results show that this method is promising because the basic graph

structures of the lexical entries are much simplified to demonstrate the comparable performance, 87 and 84 percent respectively, with that of our former system. Now the dictionary is under finer adjustment, and we are processing a large number of sentences of 10 or more speakers.

The BBN group developed a similar graphic representation of lexical entries and the dictionary expansion system. LISP functions are introduced in order to examine whether or not some phonological rules could apply in a particular context at the stage of the dictionary expansion from the orthographic description of a word. But the constructed dictionary does not contain procedural representations any more. The lexical matching of the BBN system against the output of acoustic analyzer is based on the probability calculation.

Instead we introduced procedural representation into the lexical entries themselves so that intra- and inter-word phonological transformations are treated conditionally at the lexical matching phase. Although the descriptive power of variety of pronunciation is almost the same, one could include any tests of global context, prosodic information or finer acoustic cues dependent of individual words.

Considering that the use of prosodic information has been expected, our method is well-suited for the purpose, while those systems which adopted the uniform algorithm of lexical matching, such as the Dynamic Programming, deal the problem poorly. Ours has a disadvantage that the dictionary construction is troublesome, but we consider that more information dependent of individual words must be utilized to achieve a higher performance.

REFERENCE

- [1] Y.Niimi, Y.Kobayashi, et al., The Speech Recognition System of 'SPOKEN-BASIC-1', Jour. Inform. Process. Soc. of Japan 18:5 (1977) 453-459 (In Japanese).
- [2] Y.Niimi and Y.Kobayashi, A Voice-Input Programming System Using BASIC-like Language, In Proc. IEEE ICASSP, Camelot Inn, Tulsa, OK, April, 1978, pp. 425-428.
- [3] W.A.Woods, Transition Network Grammars for Natural Language Analysis, Comm. of ACM 13:10 (1970) 591-606.
- [4] W.A.Woods, Speech Understanding Systems: Final Report 1974-1976, BBN Report 3438 (1976).