RULE DIRECTED PHONE-PHONEME MATCHING

by

Alan Cole

Memorandum No. ERL-M543

September 1975

ELECTRONICS RESEARCH LABORATORY

College of Engineering
University of California, Berkeley
94720

SUR Note # 190

RULE DIRECTED PHONE-PHONEME MATCHING

by

Alan Cole

Computational Speech and Language Group
Electronics Research Laboratory
University of California at Berkeley
Berkeley, California 94720

September 1975

ABSTRACT

One approach to machine based recognition of connected speech requires that hypothesized dictionary spellings be matched against errorful phonetic transcriptions of an utterance. A model and a rule directed phone-phoneme matching algorithm, appropriate for describing phonological variation as well as segmentation and labeling errors, are described. An experiment in which this algorithm is applied with a simple set of context-free rules suggests the usefulness of the technique, which is currently being implemented as an experimental knowledge source within the Hearsay II speech understanding system.

Research sponsored by the Defense Advanced Research Project Agency of the Department of Defense and monitored by the U. S. Army Research Office under Grant DAHC04-75-G0088.

I. INTRODUCTION

A fundamental problem for mechanical speech understanding systems is to match the measured acoustic signal to stored transcriptions of lexical items.

One approach to this problem is the two step process of (1) producing a phonetic transcription of the utterance, and then (2) comparing this transcription with entries in a phonetic (or phonemic) dictionary to determine which words were actually spoken. While such a strategy is conceptually simple, its application is complicated by at least three basic difficulties.

In the first place, the automatic production of a correct phonetic transcription is enormously difficult. Individual sounds are encoded in a continuous acoustic signal which simultaneously conveys many other levels of information--some, such as pitch or rhythm, linguistically significant; others, characteristic of the speaker's physiology or acoustic environment, linguistically irrelevant. Because the acoustic waveform is so complex, analysis techniques do not yet exist which are capable of automatically locating and classifying phonetic segments without error. Errors in the detection of boundaries between segments cause omission and spurious insertion of sounds; tentative decisions about boundaries lead to overlapping segments. In addition to erroneous boundaries, machine transcription is further confused by errors or uncertainty in the labeling of the segments themselves.

Even if speech recognition programs were able to produce perfect phonetic transcriptions, a second major obstacle exists. Speech production is a highly variable human activity, subject both to random and to systematic variation, including processes described by acoustic-phonetic and phonological rules. "Perfect" transcriptions of the same sentence, spoken by different persons or at different times by the same person, can differ appreciably. There are a large number of possible pronunciations for each word; this complicates the dictionary look-up.

Conceivably, all pronunciations for every word might be listed in the dictionary, but this would nct resolve all the problems. Pronunciations which occur only when two words are adjacent would be difficult to represent in a dictionary whose entries consisted of single words. For example, "did you" is often pronounced /D IH JH UW/, especially in rapid or casual speech, but it would be misleading to list /D IH/ or /D IH JH/ as pronunciations of "did" on an equal footing with /D IH D/, since they occur only in an appropriate right context.[1]

In short, recognition error and speaker variability combine to yield phonetic transcriptions which will rarely if ever correspond exactly to the dictionary spelling(s) of an utterance. Determination of the words comprising an utterance must be based on some procedure which allows inexact matches between the dictionary entries and the phonetic input.

[1] In this note, transcriptions are given in the two-character ARPA SUR notation.

In addition to these difficulties, a third basic problem is that reliable indicators of word boundaries occur infrequently in natural speech, so that, in the worst case, it may be necessary to match every possible word at every possible point in the utterance, a procedure which quickly becomes impractical as the size of the vocabulary increases.

In the face of these problems, it is clear that recognizing the words of an utterance is a process which is far from straightforward. It is generally felt that mechanical speech understanding systems will be able to approach the human level of performance only by incorporating knowledge from many different sources, including phonology, prosodics, syntax, semantics, and pragmatics (Newell, et al. [1973]).

In particular, the hypothesization of words may be approached from two points of view.

A bottom-up approach suggests possible words by analyzing the phonetic transcription. Often, this process is a crude but fast way of narrowing the set of possibilities. The Hearsay II system at Carnegie-Mellon University, for example, uses a dictionary which is divided into (possibly overlapping) subsets of words with a broadly defined syllable type in common (Smith [1975]). Each syllable in the phonetic input matches at most a small number of these syllable types, so that only the appropriate subsets of the dictionary need be considered further. Bottom-up methods such as this typically work best in the region of stressed vowels (where acoustic analysis performs best).

Top-down approaches suggest possible words or classes of words without direct reference to the phonetic transcription. A knowledge of syntax, for example, may suggest articles or adjectives as likely candidates immediately before a noun--again, the result is to limit the possibilities to a subset of the dictionary. A top-down method usually works best in filling in gaps left by the bottom-up approach.

The problem discussed in this note is illustrated by the Hearsay II system, in which possible words are hypothesized both by a rough, first-cut, bottom-up analysis of the phonetic input, and by top-down prediction. These possibilities must be further narrowed down by rating the hypothesized words according to a more detailed comparison of their dictionary pronunciations with the phonetic transcription.

Since the phonemic[2] spellings for the hypothesized words are known, this more detailed comparison may be viewed as an attempt to match several competing phonemic transcriptions against an errorful phonetic transcription, with the merit of a phonemic sequence determined by the goodness of the match.

This note suggests an algorithm for efficiently matching these two levels in such a way as to realistically reflect the effects of recognition error and speaker variability.

[2] For convenience, we use the term "phonemic" to indicate spellings derived from the dictionary. In fact, Hearsay II dictionary entries are expressed at the so-called "surnemic" (surface phonemic) level which represents a compromise between the phonemic and a broad phonetic level; the "phonetic transcription" is expressed at a somewhat narrower phonetic level.

II. BACKGROUND AND MOTIVATION

The basis for the suggested approach to phone-phoneme matching stems from work in spelling correction (Morgan [1970], Wagner and Fischer [1974], Lowrance and Wagner [1975]). One previous application of this technique has been in automatic correction of computer programs containing misspelled variable names or keywords. Knowledge about the probabilities of various kinds of typing errors may enable the compiler to make good guesses about the intended spelling. By executing the automatically corrected program, an extra round of editing and compilation can often be avoided.

In this model, the incorrect spelling is derived from the correct one by a series of "edit operations." Lowrance and Wagner, for example, consider the following types of operations:
(1) changing a character into some (possibly different) character; (2) inserting a character; (3) deleting a character; and (4) interchanging two characters. Each of these operations is assigned some non-negative cost; the cost of typing the correct character is assumed to be zero. The total cost of changing the correct spelling into the incorrect one is the sum of the costs of the individual edit operations.

This definition of the cost of changing one string into another provides an intuitive notion of the "distance" between the two strings. The correctly spelled word picked (as the intended word) is that word closest to the misspelled item as measured by this distance function.

Lowrance and Wagner describe a "string-to-string correction" algorithm which computes the minimum edit distance between two strings; the complexity of this algorithm is proportional to the product of the lengths of the two strings.

The application of this model and associated algorithm to the phone-phoneme matching problem is immediately suggested. As noted above, phonetic transcriptions, when compared with dictionary (phonemic) spellings, contain substitutions, insertions, and deletions due to speaker variability and machine recognition error. These differences may be modeled by edit operations, or rules, which transform the phonemic string into the phonetic string. The edit distance between a phonetic string and each of a list of candidate phonemic strings may be computed; the most likely phonemic string is that for which this distance is least.

Various difficulties, however, prevent direct application of the string-to-string correction algorithm to phone-phoneme matching. One such difficulty is that, because segmentation and labeling routines often make multiple or overlapping guesses, the phonetic transcription is really a graph rather than a string. Similarly, the possible word sequences form a graph containing many phonemic strings. Since having just two alternatives at each of N nodes of a graph gives 2^N distinct strings, a brute force approach of separately matching each phonemic string against each possible phonetic string would be impractical.

A further problem is that in the formulation given by Lowrance and Wagner, the cost associated with each type of edit operation is a constant, independent of the symbols or context involved. This assumption is certainly not valid for the phone-phoneme matching problem; for example, deletion of a stressed vowel would have a higher cost (lower probability) than the deletion of a /TH/, which is a weak and difficult to detect sound. Furthermore, the costs depend on the context. It is much more likely that a spurious /SH/ will be recognized after a stop (by confusing the aspiration with a sibilant) than that the same sound will be erroneously inserted after a vowel.

The following section describes a modification and extension of the string-to-string correction algorithm which meets these objections. Discussion of whether the model is appropriate for describing the differences between the phonemic and phonetic transcriptions is deferred to section IV.

III. AN ALGORITHM FOR PHONE-PHONEME MATCHING

This section describes an algorithm for phone-phoneme matching in the context of a system like the Hearsay II speech understanding system (Lesser _et al_. [1975]). The goal is to present only the broad outlines of the method, without formal proof. In addition, some features which are necessary or desirable in an actual implementation are discussed in a later section.

The phonetic transcription produced by Hearsay II and similar systems is, in effect, a directed graph G_P with N+1 vertices which may be labeled t_i $(0 \leq i \leq N)$, where each t_i represents the time of a boundary between two phones. Directed arcs from t_i to t_j, where $t_i < t_j$, represent phones beginning at time t_i and ending at time t_j. The start vertex at time $t_o = 0$ has in-degree zero; the final vertex $t_N = T$, where T is the duration of the utterance, has out-degree zero. An example of such a graph is shown in Figure 1.

We are interested in all paths P through G_P which start at t_o and end at t_N; each such path corresponds to a possible sequence of phones which spans the entire utterance. We will use the notation P<t> to represent that portion of the path P starting at t_o and ending at the time (vertex) t.

The phonemic spelling of the possible word sequences also results in a graph G_S at the phonemic level; the vertices v_i of this directed graph do not correspond to times (since the times or durations of phonemes proposed by top-down procedures are not

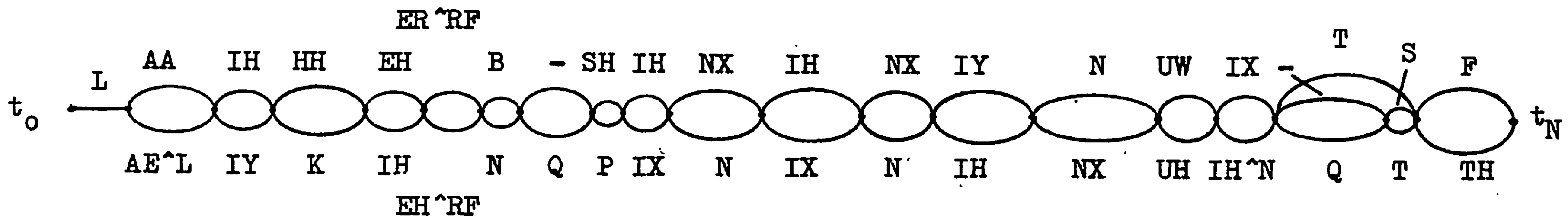

Figure 1 Graph of a machine produced phonetic transcription for the sentence "What happened in England?"

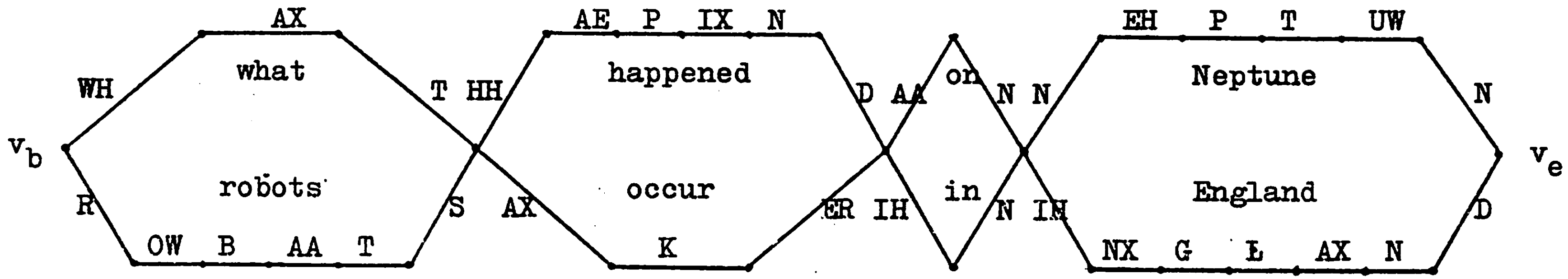

Figure 2 A phonemic graph showing the dictionary spellings of some hypothesized word sequences.

initially known). We may, however, associate the start vertex v_b with time zero, and the final vertex v_e with time T, the end of the utterance. The directed arcs of this graph, which we may label s_i $(1 \leq i \leq M)$, represent phonemes. The spellings of possible word sequences spanning the entire utterance are represented by paths S from v_b to v_e through G_S. Let S<i> represent that portion of a path S from v_b through the phoneme s_i $(1 \leq i \leq M)$; let S<0> be the null path (of length zero) starting and ending at v_b. Figure 2 shows an example of a phonemic graph.

Notice that both these graphs have a particularly simple structure. The fact that they have no self-loops or circuits will be used later.

We consider three types of mapping operations, or rules, for transforming phonemic strings into phonetic sequences:

(1) Substitution: s -> p

A single phoneme s is changed to a single phone p. The cost of this rule is given by $C_S(s,p,E)$, where "E" is used as a cover symbol to indicate the effect of the environment of s and p on the cost.

(2) Insertion: $\emptyset$ -> p

A single phone p is inserted with cost $C_I(p,E)$.

(3) Deletion: s -> $\emptyset$

A single phoneme s is deleted with cost $C_D(s,E)$.

The cost functions are non-negative, and are defined for all phonemes s, phones p, and environments E. This technicality implies, for instance, that each phoneme can be changed into any phone, no matter how unreasonable the particular mapping seems.

The unreasonableness of a rule like "AE -> TH" is reflected by assigning it a very high cost (low probability).

Superficially, these rule schemata appear to be context-free. However, they are actually context-sensitive, since the cost functions are allowed to take context into account in complex ways. In practice, various restrictions would be placed on the amount of context considered in determining rule costs.

We use these rules to represent both the errors made by the segmentation and labeling program, and the effects of random and systematic speaker variability. Variation of this latter kind is often described by low-level phonological rules considerably more complex in form than the three simple rule types listed above. While we do not assume that these simple forms of rules can completely cover low-level phonology, we do assume that they can provide a useful approximation to such coverage.

If r is an individual rule and α is an arbitrary sequence of phonemes, then $r(\alpha)$ is the resulting sequence after application of the rule r. If R is a sequence of rules $r_1 \ldots r_n$, then by $R(\alpha)$ we mean the composition of these mapping rules:

$$R(\alpha) = r_n(\ldots (r_1(\alpha)) \ldots).$$

We define $C(R,\alpha)$, the cost of such a sequence of rules acting on α, to be the sum of the costs of the individual rules.

Let D(S<i>,P<t>), the distance between a phonemic path through the phoneme s_i and a phonetic path through the time t, be defined as

$$(4) \qquad D(S{<}i{>},P{<}t{>}) = \min_{R} C(R,S{<}i{>})$$

where the minimum is taken over all sequences of rules R such that $R(S\langle i\rangle) = P\langle t\rangle$. A particular case is the distance between a phonemic path S and phonetic path P, both of which span the entire utterance:

(5) $$D(S,P) = \min_{R} C(R,S).$$

We further define:

(6) $$D(S\langle 0\rangle,P\langle 0\rangle) = 0.$$

That is, the distance between a null phonemic sequence and a null phonetic sequence is zero. This definition of distance intuitively gives a measure of the proximity of a phonemic string S to a phonetic transcription P.[1] The distance D is well-defined and exists for all pairs of sub-paths $S\langle i\rangle$ and $P\langle t\rangle$, since there is always some sequence of rules which will transform $S\langle i\rangle$ into $P\langle t\rangle$ (a trivial example is the deletion of all phonemes in $S\langle i\rangle$ followed by the insertion of all the phones in $P\langle t\rangle$).[2]

Within this framework, the phone-phoneme matching problem may be recast as the problem of finding that path S through the phonemic graph for which the distance from some path through the phonetic path is minimum. To this end, let

(7) $$H(s_i,t) = \min D(S\langle i\rangle,P\langle t\rangle),$$

where the minimum is taken over all paths $S\langle i\rangle$ from v_b through s_i in the phonemic graph, and all paths $P\langle t\rangle$ from t_o through time t in the phonetic graph.

[1] We use the term "distance" somewhat loosely. Strictly speaking, the function D is not a metric.

[2] Recall that all possible substitutions, deletions, and insertions are allowed, even though some of them may have very high costs.

If we let s_k represent a phoneme which immediately precedes the phoneme s (by convention, let $s_o = \emptyset$ be the predecessor of all utterance initial phonemes), and b(p) represent the begin time of the phone p, then the following equations allow H(s,t) to be computed recursively:

$$
(8) \quad
\begin{aligned}
H(s,t) &= \min \left\{ \begin{array}{ll} H(s_k,b(p)) & + C_S(s,p,E), \\ H(s,b(p)) & + C_I(p,E), \\ H(s_k,t) & + C_D(s,E) \end{array} \right\} \\
H(s_o,0) &= 0 \\
H(s_o,t) &= \min_{p} \left\{ H(s_o,b(p)) + C_I(p,E) \right\} \\
H(s,0) &= \min_{s_k} \left\{ H(s_k,0) + C_D(s,E) \right\}
\end{aligned}
$$

where the minima are taken over all phones p ending at time t and all phonemes s_k which immediately precede s.

Though these equations will not be proved here, it may be worthwhile to point out that they are based on the principle that if a phonemic path S is optimally matched with a phonetic path P, then each leading sub-path of S must also be matched optimally with some leading sub-path of P. In particular, an optimal match from the beginning of the utterance through the current phoneme s and the current phone p can occur in precisely one of three ways:

(a) An optimal match through a previous phoneme and a previous phone is extended by the substitution rule s -> p, which maps the current phoneme into the current phone with appropriate cost.

(b) An optimal match through the current phoneme and a previous phone is extended by an insertion rule $\emptyset$ -> p which inserts the current phone.

(c) An optimal match through a previous phoneme and the current phone is extended by a rule $s \rightarrow \emptyset$ deleting the current phoneme.

Of these three possibilities, the one which gives the lowest total cost is chosen.

If we let len(s) be the maximum length of all paths from v_b through the phoneme s, then $\mathrm{len}(s_o) = 0$ and $\mathrm{len}(s) \leq M$ for all s (since the phonemic graph G_S contains M phonemes (arcs), and has no loops). With this notation, we may now state an algorithm for finding the best path through the graph G_S, given a phonetic transcription G_P.

<u>Algorithm 1</u>

(a) Set $i = 0$.

(b) Let $X = \{ s_{i_j} \}$, $j = 1, \ldots, q$, be the set of all phonemes in G_S for which $\mathrm{len}(s) = i$. If $X = \emptyset$, terminate.

(c) Set $j = 1$.

(d) Set $k = 0$.

(e) Compute $H(s_{i_j}, t_k)$ according to equations (8).

(f) Set $k = k + 1$. If $k > N$ (the number of vertices in G_P), proceed to step (g); otherwise to step (e).

(g) Set $j = j + 1$. If $j > q$, proceed to step (h); otherwise to step (d).

(h) Set $i = i + 1$. Go to step (b).

This procedure always terminates in at most M+1 iterations of steps (b) through (h), since $\mathrm{len}(s) \leq M$. The purpose of the set X in step (b) is to ensure that when H(s,t) is computed, the previous values of H required by equations (8) have already been computed.

When the algorithm terminates, $H(s_f,T)$ gives the cost of the best path from the initial vertex v_b to each utterance final phoneme s_f. The actual sequence of phonemes comprising these best paths may be easily recovered if we record how the minimum in equations (8) is achieved.

When context is crucial to the rule cost functions, it may be necessary to save several different values of $H(s,t)$, each corresponding to a different context of the phoneme s and phonetic boundary time t.[3] The minimum in equations (8) is then also taken over all such different values of H.

Because $H(s,t)$ must be computed and at least temporarily stored for each of M+1 phonemes s and N+1 phonetic boundary times t, both the storage and time required by this algorithm are proportional to $G(M+1)(N+1)$, where G is a factor reflecting the number of different contexts relevant to determining rule costs. This value will grow exponentially as the amount of required context increases.

It is apparent that the amount of context used by the rule cost functions is crucial in determining the algorithm's efficiency. Consequently, it is probably best not to fix beforehand the width of the context considered, but rather to dynamically compute the amount of context needed at any point on the basis of rules which are applicable at that point. In this way,

[3] Actually, the different phonetic contexts could probably be safely ignored--these contexts will all be similar (at a given point in the phonetic graph) since they represent the different guesses of the segmentation and labeling routines at what is really a single phonetic context.

the penalty of a large context is paid only in those regions where it is required.

It should be noted that the procedure described above always finds the best (lowest cost) match of a phonemic sequence spanning the entire utterance with a phonetic sequence by considering matches of each utterance initial sequence of phonemes against all possible utterance initial phonetic sequences, no matter how unlikely these matches may be. For example, $H(s_f,0)$, where s_f is an utterance final phoneme, is calculated. But this represents the cost of deleting <u>all</u> the phonemes; the only way of proceding from this point to a match across the entire phonetic graph is to insert all the phones by rules of the type $\emptyset$ -> p. The fact that such a mapping of the phonemic sequence onto the phonetic graph, while mathematically possible, is so phonologically unrealistic suggests two possible modifications of the algorithm, which we will describe informally.

<u>Algorithm 2</u>

(a) Steps (d) through (f) of algorithm 1 are modified so that instead of computing $H(s_{i_j},t_k)$ for $k = 0, \ldots, N$, the function is calculated only for $k_{min} \leq k \leq k_{max}$, where the range of values is heuristically chosen to be "phonologically reasonable."

(b) Once some complete match has been found (with, say, a total cost of W), $H(S_{i_j},t_k)$ is computed for those times not previously considered. But now, whenever the total cost of any partial match equals or exceeds W, it need not be examined further, since it cannot possibly lead to a better complete match than that already found.

Though we have left unstated the exact details of this algorithm, the essential idea is to quickly find a (possibly suboptimal) complete match so that partial matches which are not as good can be pruned. This can always be done in such a way that the optimal complete match will be found; bookkeeping is somewhat more complex than for algorithm 1, but both space and time requirements are decreased.

A further modification is possible, giving a third algorithm.

Algorithm 3

As in algorithm 2, $H(s_{i_j}, t_k)$ is calculated only for "reasonable" values of k. However, the re-evaluation procedure suggested in part (b) is omitted entirely.

This procedure matches phonemic sequences only against those portions of the phonetic graph considered reasonable or likely. It is consequently not guaranteed to find the optimal match (if such a match proves, after all, to be "unreasonable"), but its space and time characteristics are better than for either of the previous two algorithms.

Because these algorithms find the hypothesized phonemic sequence and errorful phonetic sequence which are most closely matched under a set of phoneme to phone mapping rules, we term this technique "rule directed phone-phoneme matching."

This technique may also be viewed as a dynamic programming search through all possible partial matches. It is therefore not surprising that it is similar to certain well established algorithms of dynamic programming.

For example, we may construct a graph whose nodes (s,t) correspond to partial matches between a phonemic sequence through s and a phonetic sequence through time t. Three arcs from each node represent the application of the three types of rules to reach incrementally more complete matches. To each arc we assign a "length" equal to the cost of the associated rule. Finding the best match is then equivalent to computing the shortest path between (s_o,t_o) and (s_f,t_N) in this graph, a problem on which there exists an extensive literature (Dreyfus [1969]).

Another algorithm, based on a somewhat different point of view, is given by Bahl and Jelinek [1975]. They consider a phonetic sequence Y and a series of possible dictionary spellings X. Their "rules" are not applied at run-time, but are instead encoded for each dictionary sequence X in a Markov chain obtained by concatenating probabilistic finite state machines for each input symbol x in X.

In order to determine the most likely sequence X, they must compute the probability of Y given X. This they accomplish by a procedure similar to the matching technique we have described.

The principle difference between these two techniques is the way in which context is handled. Because Markov chains are memoryless, the influence of context in the Bahl and Jelinek model is treated by writing probabilistic finite state machines for each point in the n-product space x^n of input symbols, rather than just for each single input symbol x, where n is the maximum amount of context required. Consequently, this method becomes exponentially more complex as the amount of context increases.

On the other hand, the rule directed matching technique, while it is forced to apply the mapping rules anew for each match, is thereby also permitted to dynamically adjust the amount of context considered as a function of applicable rules. The complexity of this method also increases exponentially, but only with the average amount, rather than the maximum amount, of context required.

IV. IMPLICATIONS FOR A COMPUTATIONAL MODEL OF PHONOLOGY

In the previous section it was assumed that deviations between expected and machine produced transcriptions of an utterance could be adequately explained by three particularly simple classes of rules. In this section, this assumption and its implications are examined more closely.

As previously indicated, we assume that pronounceable lexical base forms are transformed into (perhaps erroneous) transcriptions of surface forms by the repeated application of three types of rules: (1) the substitution of a single surface segment for a single base segment; (2) the insertion of a surface segment; and (3) the deletion of a base segment. For each such rule, functions are defined which give the "cost" of that rule's application, based on the context.

We would like these rules to account for two separate sources of variation between the phonetic transcription and its corresponding base pronunciation. The greatest source of variation in current systems is probably the automatic segmentation and labeling of the acoustic input. Compared with the careful hand transcriptions of trained phoneticians, machine transcriptions of connected speech are still very poor.

A second source of variation is the probabilistic low-level phonological rules which alter the pronunciation of an utterance. The pronunciation of a word, even by a single speaker, may vary markedly depending on the phonological context, speech rate, speech style, etc.

We combine these two sources of variation, not because they are fundamentally the same, but because they are both manifested in the same way within a speech recognition system, as deviations of the phonetic transcription from a nominal base pronunciation.

Under the model proposed here, the base alphabet and the surface alphabet are distinct and non-overlapping. If, for ease of discussion, we assume that the base alphabet is phonemic and the surface alphabet phonetic, then this model implies that each symbol in the phonetic representation is derived by the application of some rule. The derivation of a phonetic [T] from a base /T/, for example, requires the rule

/T/ -> [T].

A second assumption implicit in the way these rules are used is that phonetic transcriptions are derived from phonemic base forms by the simultaneous application of rules to these base forms. As an immediate consequence, it is unnecessary to postulate any intermediate levels between the base and surface levels. It is also impossible to write a rule which applies to its own output, or to the output of any other rule.

While it is perhaps not unreasonable to believe that this model can account for variation of the first type, machine error, its adequacy in explaining phonological variation is far from clear. Our model of rules and their application makes two strong claims about low-level phonology which run counter to much current thought on the subject. It is therefore appropriate to offer some justification for our use of this model.

The first suspicious claim is that low-level phonological phenomena may be described by rules of substitution, insertion, and deletion of single segments only. With only these types of rules, it becomes difficult to express more complex phenomena involving the rewriting or creation of two or more segments--rules such as the interchange of two segments, the merging of two segments into a single segment, or the derivation of two surface segments from a single base segment. What we question is not the ability of the simpler rules to generate the desired output (since they can derive any phonetic output), but whether they are appropriate for the task.

An example of the second of the more complex forms mentioned above, taken from Oshika _et al_. [1975], is:

(1) $\mathrm{R\ AX} \rightarrow \mathrm{ER}\ /\ C_o \_\_ C_o \begin{bmatrix} V \\ +\ \mathrm{stress} \end{bmatrix}$.

This rule explains pronunciations like [IH N T ER D AH K SH AX N] instead of the dictionary form /IH N T R AX D AH K SH AX N/ for "introduction."

Now consider the following pair of rules, both of the simple form required by the model proposed here:

(2) $\mathrm{AX} \rightarrow \mathrm{ER}\ /\ C_o\ \mathrm{R} \_\_ C_o \begin{bmatrix} V \\ +\ \mathrm{stress} \end{bmatrix}$

(3) $\mathrm{R} \rightarrow \emptyset\ /\ C_o \_\_ \mathrm{AX}\ C_o \begin{bmatrix} V \\ +\ \mathrm{stress} \end{bmatrix}$.

It should be noted that the context in these rules is at the base, or "phonemic" level.[1] These two rules generate the desired output

[1] In the suggested model, the context would be encoded in cost functions specifying low cost (high probability) in the appropriate contexts, and high cost otherwise. To simplify the presentation, we describe these rules in a more traditional format.

(with some cost). However, since either rule can apply alone, they also generate the undesired outputs [R ER] and [AX] for the base sequence /R AX/. If the costs of these undesirable outputs were significantly higher than that of the desired output, we would be satisfied. But it is impossible to define independent cost functions for these two rules so that either one applying alone has high cost, while both applying together have a lower cost (since the cost of applying both rules is by definition the sum of the individual non-negative costs).

This inherent difficulty could be resolved in several ways. Most simply, such types of phonological processes could be ignored. This alternative would only be appropriate, of course, if the influence of such rules on pronunciation proved to be statistically insignificant.

A second approach would be to change the model to include these more complex types of rules. This is possible as long as the rules are strictly local, with a single rule never rewriting a string of length greater than some fixed constant (general "tree transformations," for example, would not be admissible). However, this extension of the model would make equations (8) of the previous section considerably more complex. This alternative, then, would be attractive only if such rules are widespread and if they significantly affect variation in pronunciation.

But a third solution, suggested by George Lakoff (private communication), is possible if the notion of "context" implied by rules (2) and (3) is broadened. Rule (2) may be left unchanged, but rule (3) may be rewritten:

(4) $$R \rightarrow \emptyset \begin{array}{l} / \; C_o \; \_\_ \; AX \; C_o \left[\begin{array}{c} V \\ + \text{stress} \end{array} \right] \\ /_s \; \_\_ \; ER. \end{array}$$

In this formulation, there are two separate contexts. The first context is the same as in rule (3), and refers to the base form, but the second context (indicated by "$/_s$") refers to the surface form, and requires that an /ER/ be present at the phonetic level immediately following the current position. Consequently, rule (4) cannot apply unless rule (2) also applies, except with very high cost. The net result is to predict either of the two acceptable outputs (the original /R AX/ or /ER/) with relatively low costs, while making both of the undesired outputs extremely expensive.

It would be interesting to compare these three alternatives to handling more complex rules in an actual speech recognition system.

But a more drastic claim of the model described in the preceding section is that rules cannot apply to their own output, so that the ordering of rules is immaterial. While such a claim is tenable on purely formal grounds, it is generally rejected in current literature (see, for example, Anderson [1974]).[2] Allowing rules to apply to their own output and to be ordered permits rules to be formulated more concisely and naturally, and provides explanations for some dialectal phenomena on the basis of different orderings of the same set of rules, though the necessity of ordering rules has also created labyrinthine problems of its own.

[2] See, however, Lakoff and Thompson [1975], where similar claims for intermediate states in syntactic derivations are suggested.

We do not here dispute that various rule ordering principles can be useful and informative in many situations. Instead, we are attempting to push a deliberately simplistic model to its limit by claiming that a simple set of non-interacting rules can approximate the actions of more traditional types of rules. The computational model of phonology proposed here is intended to be simple enough for use in actual speech recognition systems, yet rich enough to adequately describe most low-level phonological variation.

V. A SIMPLE PHONE-PHONEME MATCHING EXPERIMENT

In order to test the practicality of the model of low-level phonology and machine recognition error described in preceding sections, algorithms 1 through 3 have been implemented in a program which operates in the environment of Carnegie-Mellon University's Hearsay II speech understanding system, though it is not a part of Hearsay II.

Input to this program consists of a file containing a phonetic transcription prepared off-line by the segmentation and labeling modules of Hearsay II, and a file containing a manually prepared graph of surnemes ("surface-phonemes") describing the hypothesized word sequences for the utterance. The surneme to phone mapping rules are also read from a file.

The program then applies any one of the three algorithms described in section III, determining the paths through the surnemic graph and the phonetic graph which are most closely matched according to the given set of rules. This best match is displayed in a format like that of figure 3, which shows the best match between the phonetic graph of figure 1 and the surnemic graph of figure 2.

In the remainder of this section, we report on the first of a series of experiments designed to explore some of the assumptions and implications of the underlying model.

Best Match for Utterance # 26 'What happened in England?'

	Surneme	Phone	Time	Rule Cost	Total Cost
what	WH	L	25: 29	18	18
	Ø	AA	29: 36	82	100
	AX	IH	36: 41	33	133
	T	K	41: 49	65	198
happened	HH	IH	49: 54	18	216
	AE	ER^RF	54: 59	78	294
	Ø	N	59: 61	60	354
	P	-	61: 67	33	387
	Ø	SH	67: 68	65	452
	IX	IH	68: 71	18	470
	N	N	71: 79	18	488
	D	Ø	79: 79	46	534
in	IH	IH	79: 88	20	554
	N	N	88: 94	18	572
England	IH	IH	94: 103	20	592
	NX	N	103: 114	16	608
	G	UW	114: 119	85	693
	L	Ø	119: 119	46	739
	AX	IX	119: 124	40	779
	N	T	124: 135	90	869
	D	TH	135: 144	65	934

Figure 3: Best match of the surnemic sequence for "What happened in England" against its phonetic transcription. The begin and end times for phones are shown in centiseconds.

In this initial investigation, we attempt to measure the improvement in matching obtained by use of insertion and deletion rules in addition to phone-surneme substitution rules. This experiment was partly motivated by one version of Hearsay II which employed a phone-surneme matching strategy in which insertions and deletions were not explicitly represented as separate processes.

The rule "cost functions" used in this test were extremely simple; they were represented by a cost matrix C in which the cost of matching a surneme s with a phone p was given by C[s,p], the cost of inserting p by C[Ø,p], and the cost of deleting s by C[s,Ø]. Thus, the rule costs were independent of context--the cost matrix is essentially an ordinary confusion or similarity matrix which includes insertion and deletion frequencies.

The original cost matrix was based on phone-surneme similarities computed by the Carnegie-Mellon SUR group on the basis of actual confusions encountered in 33 sentences read by a single speaker.

In this original cost matrix CØ, insertion and deletion costs (not included in the original CMU data) were represented by two separate constants, chosen to approximate the overall likelihoods of insertion and deletion. That is, the cost of deleting a surneme was the same for all surnemes; likewise, the cost of inserting a phone was the same for all phones.

Using a technique suggested by Jelinek, Bahl, and Mercer [1975], the true costs were then estimated. The matching program was presented with the actual machine-produced phonetic transcription of each utterance, and with a surnemic graph which described only the correct sequence of surnemes. Used in this way, the program simply finds the best matching of a known surnemic sequence to a given phonetic graph.

This procedure was performed for the 33 utterances using the original cost matrix. On the basis of the resulting matches, three new cost matrices were computed, one based on utterances 1-16, another on 17-33, and a third based on all 33 utterances. This procedure was iterated, this time using the new matrices, and resulted in a final set of three cost matrices: (1) C1-16, based solely on utterances 1-16;[1] (2) C17-33, based on utterances 17-33; and (3) C1-33, based on the entire set of data.

The correct surnemic spellings were then augmented by adding, for each correct word, a random selection from a list of phonetically similar "distractor words."[2]

The matching program was reapplied to these augmented surnemic graphs, using, in turn, the original cost matrix CØ (in which insertion and deletion costs were constant), and the three derived matrices C1-16, C17-33, and C1-33. For each of these four cases, table 4 shows the number of words correctly picked as well as the number of utterances in which all words were correctly picked. Performance is shown separately for utterances 1-16 and for 17-33; totals for the entire 33 are also shown.

Thus, of the 229 word tokens in all 33 sentences, the original cost matrix found 184 or 80.3 % correctly, while the derived cost

[1] Except that the original similarity matrix obtained from CMU was based on all 33 utterances.

[2] The total number of words in the vocabulary was 57.

		Cost Matrix			
utterances		CØ	C1-16	C17-33	C1-33
1-16	utterances correct	7	11	7	10
	words correct (of 108)	85	101	92	100
17-33	utterances correct	5	3	10	11
	words correct (of 121)	99	97	113	114
1-33	utterances correct	12	14	17	21
	words correct (of 229)	184	198	205	214

Table 4: Performance of phone-surneme matching on 33 utterances with cost matrices
(a) CØ based on all 33 utterances, but constant insertion, deletion costs
(b) C1-16 computed from just the first sixteen utterances
(c) C17-33 computed from the last seventeen utterances
(d) C1-33 computed from all 33 utterances

matrix C1-33 found 214, or 93.4 %, against expected chance performance of 50 %.

Not unexpectedly, the use of measured insertion and deletion costs produced a substantial improvement over the use of overall averages for these costs. Student's t test shows that the better performance of the derived cost matrix C1-33 compared with that of the original matrix CØ is significant at a confidence level well in excess of 99 %. In fact, there is a tendency for the derived matrices based on only half the data to perform better than the original matrix (based on all the data), though this difference is significant only at the 70 % level.

We conclude that, for the current segmentation and labeling strategies in Hearsay II, use of deletion and insertion frequencies, even if averaged over all contexts, can provide significantly better recognition rates than use of confusion frequencies alone.

Though the experiment just described involves a particularly simple model in which rule costs are independent of context, it still raises some interesting questions which we have not answered here. For example, cost matrices with supposedly realistic entries for insertions and deletions were obtained in two iterations of a procedure which started with a matrix in which insertion and deletion costs were constants. We may ask how fast this procedure converges (or whether it converges at all). Would a single iteration have been sufficient? How much better would three iterations have been?

Or, again, results were obtained for hypothesized word sequences which always contained the right number of words, with exactly one wrong word for each correct word. How would performance have fallen off with more realistically complex graphs of hypothetical word sequences?

Although the answers to such questions would be informative, we feel that a more important direction for further study is the use of context in determining accurate rule costs. How much context is required at the surnemic and phonetic levels to satisfactorily describe the observed variations and differences between these levels? And what price in algorithm complexity is paid for the increased accuracy afforded by these more precise rules?

Especially useful would be a procedure which automatically derived the costs of rules as a function of their context, perhaps by iteratively refining rough initial cost estimates in the same way that the context-free cost matrices were derived.

Besides its direct application in a recognition program, such a procedure would permit an easy assessment of the relative importance of phonological variation versus recognition error. Two sets of rule costs could be measured; one, based on the differences between dictionary pronunciations and careful hand transcriptions, would reflect true speaker variability; the other, based on the differences between hand and machine transcriptions, would describe the effects of segmentation and labeling error. These two sets of rules, used separately and in combination, could help in deciding how important speaker variability is in the face of the high recognition error rates of current systems. Similar methods might also be used to quantitatively measure the magnitude of inter-speaker differences.

VI. IMPLEMENTATION OF PHONE-PHONEME MATCHING IN HEARSAY II

At present, we are in the process of implementing the rule directed phone-phoneme (or phone-surneme) matching technique as a knowledge source within the Hearsay II speech understanding system. As previously described, this technique is not immediately applicable within Hearsay II for several reasons.

For one thing, the algorithm matches a phonetic graph against a hypothesized phonemic graph, both of which are assumed to be present in their entirety before the matching begins. This may well not be true for the phonetic graph if the system is operating, or simulating operation, in real time. In this case, the phonetic graph is extended to the right as each new portion of the acoustic input is analyzed; having to wait until the entire utterance has been heard before beginning a match can destroy much of the motivation for real time processing. But the phonemic graph is even less likely to be present and complete when the matching begins, for the very process of hypothesizing new words almost requires, especially for top-down prediction, that some previously hypothesized words already be rated.

Furthermore, good matches are determined by extending previously calculated sub-matches strictly to the right. Consequently, a "gap" in which either the segmentation and labeling routines or the word hypothesization routines are unable to make any guesses will completely block the matching algorithm.

An additional problem is that while the set of hypothesized word sequences contains at most one correct sequence, it may be expected to include many incorrect sequences. When two or more matches "merge" at some common node in the phonemic graph, the worst matches can be discarded, but, until that time, as much effort is expended on exploring a hopelessly bad match as on a promising one.

Clearly, these problems must be dealt with if the phone-phoneme matching algorithm is to prove viable in a real speech recognition system.

The following principle makes possible a large step in this direction.

Assumption

> The variation in a phonetic transcription on one side of a stressed vowel or pause, whether due to speaker variability or recognition error, is independent of the context on the other side of that stressed vowel or pause.

We will use the term "anchor point" to refer to phonetic pauses or stressed vowels, and to the beginning and end of an utterance.

To the extent that this assumption is true, we are justified in breaking up the phonetic and phonemic graphs into sections between anchor points, and computing best matches separately within each section. The overall best match is then obtained simply by concatenating the best matches from each section. Not only is such a procedure more efficient than the original algorithm, but it helps to solve the problems associated with the practical use of the algorithm.

A further modification is that whenever a match has been extended to the end of a word, that word is rated according to the cost of that portion of the match spanning the word. If this match later turns out to be sub-optimal, then the word's rating is adjusted accordingly. But in any case, the initial rating provides a lower bound for the word's eventual rating which can be used, in the absence of any better information, to focus the efforts of the top-down word prediction modules.

A third change is that, although the matching algorithm was previously described in terms of left to right processing, both right to left and left to right matches can be determined by essentially the same algorithm.

The implementation of the matching technique within Hearsay II combines all these modifications, so that it is no longer necessary to wait until the entire utterance has been input and all word sequences hypothesized before matching can begin. The tentative rating of words during the determination of a best match triggers further activity by the word hypothesizing modules. And by extending matches both to the right and to the left of each anchor point, it is possible to surround gaps with words whose ratings are approximately known, so that top-down predictions of words occurring within the gap can be made more easily.

By totally abandoning matches when their total cost exceeds some threshold (adjusted for the length of the match), it is also possible to reduce the amount of time spent in rejecting bad matches, though this also introduces the possibility of overlooking what may actually be the best match.

These techniques are currently implemented in an experimental Hearsay II module named MAP, which actually contains three separate "knowledge sources." The first two are extremely simple: PAUSE identifies pauses at the phonetic level and hypothesizes corresponding word boundaries at the surnemic level; ANCHOR identifies stressed vowels (located by another module within Hearsay II as described by Smith [1975]) and pauses as anchor points. The surnemic segments or boundaries corresponding to these phonetic positions are also marked, thereby effectively dividing the utterance into non-overlapping sections.

The third, most complex, component of the module, named MATCH, determines the best sequence of surnemes within each such section of the utterance, using algorithm 3 modified as suggested above.

Presently, a cost matrix as described in section V is used to specify the rule costs.

The testing of this module is still in a very early stage. It has not yet been completely integrated into the Hearsay II system; the few test cases so far evaluated have used the actual phonetic transcription produced by Hearsay, but the hypothesized word sequences have been manually selected. We hope to reach a point soon at which use of the module will become routine, permitting detailed comparisons of its performance with other phone-surneme matching techniques currently used in Hearsay II.

REFERENCES

Anderson, Stephen R. (1974). *The Organization of Phonology*. New York: Academic Press.

Bahl, Lalit R. and Frederick Jelinek (1975). "Decoding for Channels with Insertions, Deletions, and Substitutions with Applications to Speech Recognition," *IEEE Transactions on Information Theory* IT-21, 4 (July 1975), pp. 404-411.

Dreyfus, Stuart E. (1969). "An Appraisal of Some Shortest-Path Algorithms," *Operations Research* 17, 3 (May-June 1969), pp. 395-412.

Jelinek, Frederick, Lalit R. Bahl, and Robert L. Mercer (1975). "Design of a Linguistic Statistical Decoder for the Recognition of Continuous Speech," *IEEE Transactions on Information Theory* IT-21, 3 (May 1975), pp. 250-256.

Lakoff, George and Henry Thompson (1975). "Introducing Cognitive Grammar." To appear in *Proceedings of the First Annual Meeting of the Berkeley Linguistics Society*. University of California at Berkeley: Institute for Human Learning.

Lesser, V. R. *et al.* (1975). "Organization of Hearsay II Speech Understanding System," *IEEE Transactions on Acoustics, Speech, and Signal Processing* ASSP-23, 1 (February 1975), pp. 11-24.

Lowrance, Roy and Robert A. Wagner (1975). "An Extension of the String-to-String Correction Problem," *JACM* 22, 2 (April 1975), pp. 177-183.

Morgan, Howard L. (1970). "Spelling Correction in Systems Programs," *CACM* 13, 2 (February 1970), pp. 90-94.

Newell, A. *et al.* (1973). *Speech Understanding Systems: Final Report of a Study Group*. North-Holland.

Oshika, B. T. *et al.* (1975). "The Role of Phonological Rules in Speech Understanding Research," *IEEE Transactions on Acoustics, Speech, and Signal Processing* ASSP-23, 1 (February 1975), pp. 104-112.

Smith, Richard (1975). POMOW: HSII Word Hypothesizor. Sur Note # 173. Carnegie-Mellon University (June 1975).

Wagner, Robert A. and Michael J. Fischer (1974). "The String-to-String Correction Problem," *JACM* 21, 1 (January 1974), pp. 168-173.