The lexical access component of the CMU continuous speech recognition system.

Alexander I. Rudnicky, Lynn K. Baumeister, Kevin H. DeGraaf, and Eric Lehmann

Department of Computer Science, Carnegie-Mellon University, Pittsburgh, Pennsylvania 15213.

Abstract

The CMU Lexical Access system hypothesizes words from a phonetic lattice, supplemented by a coarse labelling of the speech signal. Word hypotheses are anchored on syllabic nuclei and are generated independently for different parts of the utterance. Junctures between words are resolved separately, on demand from the Parser module. The lexical representation is generated by rule from baseforms, in a completely automatic process. A description of the various components of the system is provided, as well as performance data.

This paper describes the lexical access system under development at Carnegie-Mellon University. The design of the hypothesizer is based on the following principles:

- ***Words can be generated bottom-up with a very high degree of accuracy.*** Given a sufficiently accurate transcription of the speech signal, it is possible to use a completely bottom-up paradigm to drive word recognition, without assistance from higher-level constraints, such as those that might be provided by a narrowly defined task, or restrictive grammar.
- ***Multiple knowledge sources are necessary for generating high-quality word hypotheses.*** The information contained in a phonetic transcription is of itself insufficient to guarantee high accuracy, additional constraints on interpretation, either derived from alternate analyses of the signal, or from stored knowledge about speech characteristics are necessary for accurate hypothesizing.

The word hypothesizer produces lexical hypotheses using the phonetic label lattices produced by the Acoustic-Phonetic component of the system [1]. Figure 1 presents a schematic diagram of the hypothesizer module. The principal functional components of the word hypothesizer are the following:

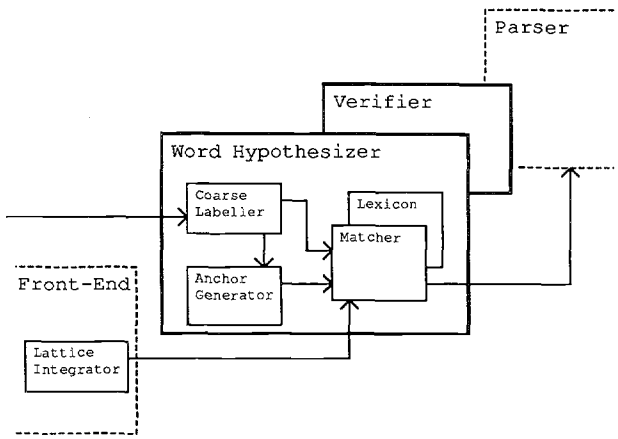
- **Matching Engine:** The matcher generates a lattice of word hypotheses. A modified beam-search algorithm is used to match a phonetic transcription against a lexicon stored in the form of a phonetic network.
- **Anchor Generator:** The matcher does not attempt to match words at all possible positions in an utterance, as might, for example, a two-level DP algorithm. Rather, the anchor generator uses a coarse segmentation of the speech wave to locate syllable nuclei and to define likely word regions ("anchors").
- **Coarse Labeller:** It is capable of producing a robust segmentation of the speech signal into silence, noise and vocalic regions. Coarse labels are used both to locate syllable nuclei and as a secondary source of information by the matcher.

In addition to the above components, the lexical access system also includes a **Phonetic Lattice Integrator** and a **Juncture Verifier**.

The **Phonetic Lattice Integrator** combines and transforms the independently generated information contained in the stop, closure, vowel, and fricative lattices produced by the Acoustic-Phonetic labelling component. The actions performed by the Phonetic Lattice Integrator include the adjustment of boundaries, the resegmentation of overlapping segments, and the combination of label probabilities from different lattices.

The role of the **Verifier** is to process word-juncture verification requests generated at the Parser level. The Verifier analyses junctures between words and indicates to the Parser whether the words in question can form a phonetically acceptable sequence.

Figure 1: Word Hypothesizer system diagram



1 Matching Engine

Words are hypothesized by matching an input sequence of labels against a stored representation of possible pronunciations, the *lexicon*. The matching algorithm makes use of both a phonetic lattice and a coarse lattice. The network search algorithm used in the current system is based on the beam search algorithm, but has been substantially modified to deal with the particular demands of the current task.

Beam-search is a modified best-first search strategy that extends paths with scores within some window of the global best score. The width of this window (the "beam") controls the severity of pruning applied to the search. The principal difference between a conventional beam-search (as implemented, e.g., in the HARPY system [3]) and the current algorithm is the ability to simultaneously search paths of different lengths. Although search tree is expanded segment by seg-

CH2396-0/87/0000-0376 $1.00

ment (i.e., is time-locked), paths may begin at a number of separate locations in the anchor region (see below). Because of the resulting differences in path lengths, the bounds of the beam cannot be calculated in a simple fashion. The solution used is to normalize all path scores by their duration.

The size of the search tree is controlled in two ways, by modifying the width of the beam and by altering the score of a given path through the use of penalties.

Beam width is calculated dynamically at each ply and is based on a pre-set width modified by a value based on the size of the expansion stack generated at the preceding ply. The effect is to relax pruning when there are few nodes on the stack and to tighten it when the stack begins to grow excessively large. One practical consequence of this is to allow paths that initially have poor scores to survive long enough to accrete positive evidence. Another consequence is to permit more severe pruning later in the search when the number of path is typically the greatest. Dynamic beam adjustment speeds the algorithm up by 39%, and reduces the depth of the output lattice by 18%, while maintaining match accuracy.

In addition to pruning based on beam width, the system uses several other strategies to control the size of the search tree. Since search progresses uniformly through successive segments, paths that pass through the same node in the network at the same segment ("collisions") are compared, and only the best path is kept (work with HARPY has shown that although this is a sub-optimal strategy, it nevertheless, in practice, produces near-optimal network traversals, at substantial savings in search effort).

Two additional pruning factors come into play through their ability to modify the cost of a path and thereby place it outside the search beam.

The first of these is a duration range associated with each phonetic label in the lexicon. Paths that remain in a particular state (phone) for either shorter or longer than the characteristic range for that phone incur a penalty. For example, the duration range for a /b/ is [3 30], based on the observation that /b/ bursts typically do not exceed 20ms. the constraint for an /s/ is [50 250], again based on the observation that /s/ phones are typically at least 40ms in duration. Similarly, the duration constraint provides a different range for an /ax/ as opposed to a diphthong, such as /a^{y}/. Exiting a state either too early or too late incurs a penalty, this penalty is added to the path score.

A second type of penalty is assessed when the coarse class of a phone mismatches that provided by the coarse labeller. The assumption here is that if the two types of label do not match, an error is likely. Again, the penalty added to the path score makes it a candidate for pruning. If the match is already poor, this penalty hastens its pruning. In fact, this penalty is most useful for rapidly terminating paths that wander across category boundaries, for example, remaining in a vocalic state when the segments have become non-vocalic. In the current implementation, enforcing cross-lattice consistency reduces the size of the search by a factor of about 3. If consistency were absolutely enforced (i.e., inconsistency results in immediate pruning) search would be reduced by a factor of 6-7 though with a loss in accuracy.

The calculation of the path score is performed according to the following formula:

$$\frac{\sum_{i=1}^{n} d_i(k \log p_i)}{\sum_{i=1}^{n} d_i} + \frac{\sum_{i=1}^{n} a_i P_D}{\sum_{i=1}^{n} d_i} + \frac{\sum_{i=1}^{n} d_i P_L}{\sum_{i=1}^{n} d_i} + q(S_E - S)$$

The formula consists of three terms: the phonetic score, the duration penalties, and the lattice mismatch penalties; n is the length of the path. The phonetic score consists of the following: d_i is a segment duration, p_i is a label probability, and k is a scaling factor (a computational convenience)

The duration penalty consists of a_i, the amount of discrepancy, and P_D, a system parameter controlling the degree of penalty. The lattice penalty consists of a system parameter, P_L, scaled by the duration of the segment, d_i. Normalization is necessary, as paths of different length need to be comparable. The final term in the equation represents a state shortfall. Each hypothesis in the lexicon is required to match a minimum number of core phonetic states. Matching less than this number implies that word has been severely reduced, a condition which is penalized in the current system.

2 The Lexicon

The lexicon is stored in the form of a phonetic network. The process of creating a net is as follows: For the chosen vocabulary, a set of base-form pronunciations is obtained. The sources of pronunciations that have been made use of include the following: lookup in an on-line phonetic dictionary, such as the Shoup dictionary, the generation of pronunciations using a letter-to-sound compiler (the MITalk system), or direct construction. Each approach has its advantages and disadvantages. We have found that automatic generation as a first pass, followed by hand correction, generally produces the most acceptable result and does so in a reasonable amount of time. Baseforms are further expanded into pronunciation networks in order to take into account different possible realizations of a word, such as those due to rapid-speech phenomena and coarticulatory effects. Possible variations in pronunciation are expressed in the form of phonological rules that are applied automatically (in an off-line procedure) to the baseform pronunciation. Figure 2 shows a typical rule, governing /iy/ desyllabification. The rule-applier scans the pronunciation string for the pattern specified in the FIF portion of the rule, binding the elements of the pattern as specified. Terms headed by a "+" match 0 or more elements, which are bound to the following variable (e.g., `LeftContext`). Terms headed by ">" must match a single element, typically meeting the constraints specified in the remainder of the clause; constraints are expressed in terms of phonetic features, such as CONS (consonant) or VELAR (a place of articulation). The THEN part of the rule has two clauses, the first specifies the portion of the pronunciation string to be emitted, the second clause the portion to be rescanned with the pattern. Depending on what is put into each clause, a rule may be made to apply once, multiple times, or iteratively to a pronunciation. The current CMU lexicon is constructed using a base of over 150 rules, covering several types of phenomena, including coarticulatory phenomena and front-end characteristics. A small number of additional rules perform necessary bookeeping functions.

Expansion is performed by adding nodes and arcs to the base pronunciation through the application of phonological rules. The individual nets produced in this fashion are then merged together into a single network, the representation used by the matcher. The merge collapses common initial states to eliminate redundant matches and produces a network that fans out from few initial states into a larger number

Figure 2: A phonological rule

```
(IY-syl-loss-a
  (FIF ( (+ LeftContext)
         (> Tm1 (has CONS) )
         (> Tar (has VOWEL HIGH FRONT) (lacks LAX))
         (> Tp1 (has VOWEL))
         (+ RightContext)
         )
   THEN
        ( (LeftContext
           Tm1 (alt 'Y 'IY) Tp1)
          (RightContext))
       )
 )
```

The above rule applied to the word COLUMBIA:

```
        K AX L UH M B IY AX
---->
        K AX L UH M B (Y , IY) AX
```

of states, the penultimate states corresponding to individual lexical entries.

3 Anchor Generation

The structure of speech constrains the possible locations of words in an utterance, that is, a word may not begin or end at some arbitrary point; permissible end-points are governed by the acoustic properties of the signal. To eliminate unnecessary matches, the system uses syllable anchors to select locations in an utterance where words are to be hypothesized. The anchor selection algorithm is straightforward and is based on the following reasoning. Words are composed of syllables, syllable all contain a vocalic center (de-voiced syllables can be treated as a special-case). Word divisions cannot occur inside a vocalic center, thus all syllable and word breaks will occur in the regions between vocalic centers. The coarse labeller provides information about vocalic, non-vocalic, and silence regions, as well as information about energy dips within vocalic regions (typically corresponding to liquids, glides, and nasals). This allows the utterance to be segmented into two regions: vocalic centers and boundary regions. An anchor, as used by the matcher, consists of two anchor regions, a beginning and an ending one, separated by one or more vocalic centers, the number of centers determining the number of syllables that words hypothesized for that region should have. Figure 3 provides a schematic diagram of the anchoring process.

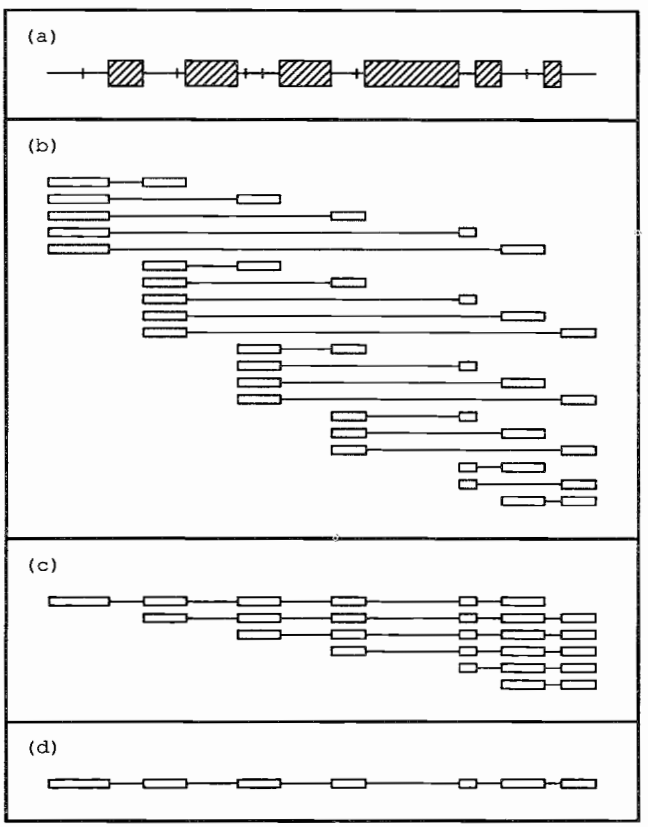
The matching algorithm allows words to begin anywhere in the beginning region (i.e., the initial state of the network is put on the stack for each phonetic segment in this region). Paths may not transition into the the network's final state until path extends into the ending region. The algorithm is implemented in such a fashion that, for a given word in the lexicon, only a single, "best" hypothesis will be generated, where best means the lowest cost traversal through the lexical network.

Anchors have been used in the system in two modes *single-anchor* and *multiple-anchor*. In the single-anchor mode, anchors of different lengths are generated and the matcher is invoked separately for each one, as shown in (b). It should be apparent that this procedure, although simple, is inefficient. There two reasons for this: The entire network is applied to each anchor, thus time is wasted trying to force, e.g., 5-syllable words into 1-syllable anchors. Second, the same region of speech is scanned repeatedly, with the results of one scan being unavailable to subsequent scans. The multiple-anchor strategy alleviates these problems, at only a slight increase in algorithm complexity, by using anchors with multiple end regions. In this case, paths for words of inherently different durations can terminate at compatible points in the anchor and are not forced into inappropriate regions. A multiple-anchor strategy reduces computation by a factor of 3, while reducing the number of hypotheses generated by 60% (inappropriate mappings of words into syllables are eliminated). A third strategy is possible, though at this time has not been implemented. This is the use of *continuous anchors*, where each inter-vocalic region serves both as an entry point and and end-region for the search (d). The advantage of a continuous anchor strategy is that it allows the simultaneous comparison of paths that span different portions of the signal. The quality of input, however, determines the success of this strategy.

4 Coarse Labeller

The coarse labelling algorithm is based on the ZAPDASH algorithm [2], modified to generate additional labels and to provide a more accurate segmentation of the signal. The coarse labeller codes the speech signal using four parameters extracted on a centisecond basis, these being peak-to-peak amplitude and zero-crossing counts for low-passed and high-passed portions of the signal (the crossover being at 1 kHz). Segments are located by seeking frames characteristic of a

Figure 3: Anchor region selection



Notes: (a) A coarse segmentation of the speech signal. The hatched blocks are vocalic centers. (b) Single anchors for the signal in (a), a total of 20 anchors. Search can begin from any segment in the onset region, must proceed through the middle, and can terminate in the coda region. (c) multiple anchors, search can begin in the first region and end at any subsequent region. (d) continuous anchors, search can begin in any but the last region and end end at any but the first region.

particular energy type using a strict criterion (an "anchor"), then expanding these into a region using a laxer criterion. In addition to the anchor-extend procedure, rules are used to apply contextual information to ambiguous regions and to perform boundary adjustment.

The algorithm currently distinguishes the following acoustic events: silence, including "true" silence and noisy silence; sonorants, including vocalic centers as well as inter-vocalic sonorant energy dips (such as nasals or liquids); a variety of aperiodic signals, corresponding to fricatives, aspirates, etc.

The algorithm is robust and speaker-independent, and operates reliably over a large dynamic range. Currently, the quality of coarse-labelling is such that less than 0.1% of syllabic nuclei are missed. A number of extra nuclei are generated, though this does not create difficulties for either anchor generation or lattice cross-checking during matching.

5 Phonetic Lattice Integrator

The phonetic labels produced by the front-end [1] are grouped into four separate lattices: vowels, fricatives, closures, and stops. Moreover, labels both within and between lattices may overlap in time. The role of the integrator is to combine these separate streams and produce a single lattice consisting of non-overlapping segments, each segment containing the information from one or more segments in the original lattices. The integrator maps the label space used by the front-end into the label space used in the lexicon. For example, the label "stop" is expanded into the appropriate set of lexical labels ([ptkbdg]). In addition, the integrator uses a confusion matrix to partition the probability assigned to a front-end label into several labels that it may be confused with, thus an input iy label will be reflected in not only the lexical IY label, but also the IH label.

The use of a confusion matrix to map the input symbol produces an improvement in accuracy, but at the cost of additional search. For the 708 word Shipping Management task, first choice accuracy goes from 32% to 42%, while the average number of states examined per word rises 2.5-fold, from 958 to 2381. We believe that the advantage of this transformation is due to the ability of the confusion matrix to capture the broad behaviour of classifier labels across different contexts and thereby supplement the probabilities generated for a given classification region (see [1])

6 Verifier

Words are hypothesized in isolation, that is, without regard to any sequential constraints between words. In this sense, the system is completely bottom-up, since no syntactic, semantic, or task constraints are brought to bear on the process of hypothesization. The resulting word lattice consequently contains many potential sequences of words. The parser [4] attempts to construct plausible sequences, but does not have the information necessary to decide whether a particular sequence is phonetically acceptable. The Verifier examines junctures between words and determines whether these words can be connected together in a sequence. The verifier deals with three classes of junctures: *abutments*, where two words join together without overlap or intervening segments; *gaps*, where the two words are separated, and *overlaps* where the words share one or more segments. In general, overlaps that involve inconsistent interpretations of the speech signal are disallowed, and gaps that contain significant speech events are also disallowed. Figure 4 shows the distribution of juncture types for the Email task (considering only correct word sequences), together with Verifier accuracy.

Figure 4: Juncture types and Verifier performance

Juncture type	Incidence	(rejection)
Abuts	51%	(0%)
Gaps	20%	(1.7%)
Overlaps	29%	(5.9%)

7 Performance

System performance was evaluated by calculating the rank of the correct word for a known anchor position. This metric is somewhat conservative, since words with the same core but with different endpoints are compared (for example, the embedded word END competes with the word SEND under the current scheme). Figure 5 gives performance for two types of input data, spectrogram reading and automatic labelling (using the September 1986 CMU system). The task is the 324 word Electronic Mail task.

Figure 5: Word Matcher performance

	Spectrogram	Automatic
1st choice	60%	32%
Top 3	83%	55%
Top 10	93%	76%

8 Conclusion

The CMU lexical access system operates as a word-spotter, generating all likely hypotheses, anchored on syllable nuclei. The design of the matching algorithm demonstrates the appropriateness of a unified matching strategy, as opposed to a strategy that uses coarse-filtering of word candidates followed by fine-grain phonetic matching: Coarse-class constraints are used as a component of the pruning strategy and do not entail the use of hard decisions implicit in, e.g., a filter design. This approach provides a maximum of flexibility to subsequent levels of processing.

9 Acknowledgement

The research described in this paper is sponsored by the Defence Advanced Projects Agency under contract N00039-85-C-0163. Views and conclusions contained in this document are those of the authors and should not be interpreted as representing official policies, either expressed or implied, of the Defence Advanced Research Projects Agency or of the United States Government.

10 References

[1] Cole, R., Phillips, M., Brennan, B., & Chigier, B. The CMU phonetic classification system. In *Proceeedings of the ICASSP*. 1986.

[2] Gill, G., Goldberg, H., Reddy, R., & Yegnanarayana, B. *A recursive segmentation procedure for continuous speech.* Technical Report, Carnegie-Mellon University, May, 1978.

[3] Lowerre, B. and Reddy, R. The HARPY speech understanding system. *Trends in speech recognition.* Prentice-Hall, 1980.

[4] Stern, R.M, Ward, W.H., Hauptmann, A.G., & Leon, J. Sentence parsing with weak grammatical constraints. In *Proceedings of the ICASSP*. 1987.