Automatic Phonetic Baseform Determination

L. R. Bahl, S. Das, P. V. deSouza, M. Epstein, R. L. Mercer, B. Merialdo, D. Nahamoo, M. A. Picheny, J. Powell

Continuous Speech Recognition Group,
Computer Sciences Department
IBM Research Division,
Thomas J. Watson Research Center
P. O. Box 704, Yorktown Heights, NY 10598

ABSTRACT

Phonetic baseforms are the basic recognition units in most large vocabulary speech recognition systems. These baseforms are usually determined by hand once a vocabulary is chosen and not modified thereafter. However, many applications of speech recognition, such as dictation transcription, are hampered by a fixed vocabulary and require the user be able to add new words to the vocabulary. At least one phonetic baseform must be assigned to each new word to properly integrate the word into the recognition system. Dictionary lookup is often unsuccessful in determining a phonetic baseform because new words are often names or task-specific jargon; also, talkers tend to have idiosyncratic pronunciations for a substantial fraction of words. This paper describes a series of experiments in which the phonetic baseform is deduced automatically for new words by utilizing actual utterances of the new word in conjunction with a set of automatically derived spelling-to-sound rules. We evaluated recognition performance on new words spoken by two different talkers when the phonetic baseforms were extracted via the above approach. The error rates on these new words were found to be comparable to or better than when the phonetic baseforms were derived by hand, thus validating the basic approach.

1 Introduction

Phonetic baseforms are the basic recognition units in most large vocabulary speech recognition systems. These baseforms are usually determined by hand once a vocabulary is chosen and not modified thereafter. However, many applications of speech recognition, such as dictation transcription, are hampered by a fixed vocabulary and require the user be able to add new words to the vocabulary. At least one phonetic baseform must be assigned to each new word to properly integrate the word into the recognition system[1]. This paper describes a series of experiments in which the phonetic baseform is deduced automatically for a new word given its spelling by utilizing actual utterances of the new word in conjunction with a set of automatically derived spelling-to-sound rules.

Most previous attempts to construct a phonetic baseform for a new word have been in the area of speech synthesis. Klatt [3] reviews various approaches to this problem; most solutions involve some set of human-derived rules combined with a dictionary lookup procedure for exceptions. The best systems seem to produce correct baseforms for 95%-98% of the words in running text. We believe that such systems are inadequate to handle the baseform construction problem in speech recognition for several reasons. First, many new words in speech recognition systems are either names or task-specific jargon (see discussion in section 4). Pronunciation of such items tend to be highly irregular and present substantial difficulties for text-to-speech systems [3]. Second, talkers tend to develop idiosyncratic pronunciations of many words, especially proper names. For example, the name *Picheny* is often pronounced *Pitch'-uh-nee* rather than *Pitch-eh'-nee* even by people familiar with the individual, and the word *asymptotic* can be pronounced with an initial *EI* rather than with an *AE* by people not terribly familiar with the word. Thirdly, the spelling of a word sometimes has very little correlation with its pronunciation - the orthography *AAA* is often pronounced *Triple-A*. Finally, human-derived rule-based systems are very labor intensive to create and often very hard to transfer from the original development site to other institutions that wish to utilize it.

It seems obvious from the above examples that proper deduction of a phonetic baseform for a new word requires both knowledge of the spelling of the word and at least one acoustic example of its pronunciation. While in theory it should be possible to deduce a baseform from the utterance alone, current speech recognition systems do not yet produce phonetic transcriptions with a high enough accuracy for this to be feasible. We follow the basic approach as described in Lucassen [4]. In particular, our goal is to find the phone string $\mathcal{P}$ to maximize

$$p(\mathcal{P} \mid \mathcal{L},\mathcal{U}) \approx \underset{\mathcal{P}}{argmax}\, p(\mathcal{U} \mid \mathcal{P})\, p(\mathcal{P} \mid \mathcal{L}). \qquad (1)$$

where $\mathcal{U}$ represents the utterance(s) and $\mathcal{L}$ represents the word spelling. This is a standard speech recognition problem in which $p(\mathcal{U} \mid \mathcal{P})$ is computed with an acoustic model, usually a Hidden Markov Model [5], and $p(\mathcal{P} \mid \mathcal{L})$ is computed with a language model. In Lucassen's work, this language model was constructed automatically from pairs of word spellings and pronunciations with the aid of a decision tree.

We have advanced Lucassen's original work in the following directions. First, the construction of decision trees has

[1] One does not necessarily require a phonetic baseform to add a new word to a recognition system [1, 2], but if the system is originally based on phonetic baseforms, the system structure is complicated by having more than one type of recognition model present.

been studied in much more detail [6, 7, 8] since the original work was performed; this allows us to construct better trees. Second, Lucassen only had the data in Webster's 7th Dictionary avaliable in incomplete form for training the decision trees; we now have access to other sources of data; e.g., the 20000 baseforms used in the Tangora [9]. We have also substantially improved our techniques for aligning phone strings agianst letter strings, a crucial component of decision tree construction (Section 2.2). Third, Lucassen used a single utterance of a word to construct a baseform; we have since found that multiple utterances of a word can be utilized to construct a more consistent baseform. Finally, in the original study, the evaluation of all baseforms was done via visual inspection and recognition performance was not measured; we will describe a series of experiments in which we attempt to evaluate the performance of the baseforms for recognition.

The structure of the paper is as follows. Section 2 describes the construction of decision trees for spelling to sound rules. Section 3 describes how a phonetic baseform is decoded using one or more utterances in conjunction with the spelling-to-sound rules. Section 4 describes a series of experiments in which the recognition performance of the baseforms is measured. Finally, Section 5 discusses the recognition performance and has suggestions for future work.

2 Decision Trees for Spelling-to-Sound Rules

Our goal is to construct a model for $p(\mathcal{P} \mid \mathcal{L})$, the probability of the phone string given the letter string. More specifically, let $\mathcal{L} = l_1, \ldots, l_n = l_1^n$ be a string of letters and $\mathcal{P} = p_1, \ldots, p_m = p_1^m$ be the string of phones corresponding to $\mathcal{L}$. We assume corresponding to each letter l_i is a string of phones which can be interpreted as the *pronunciation* of the letter. The pronunciation of letter l_i, denoted as r_i, may correspond to a single phone, a string of phones, or the null phone (i.e., it has no corresponding pronunciation in the original phone string $\mathcal{P}$. For example, Table 1 indicates the pronunciations of the letters in the word *humane*.

h	u	m	a	n	e
h	juul	m	eil	n	$\emptyset$

Table 1: Pronunciations of letters in the word *humane*.

Note that the *e* is silent, so we assign it a null pronunciation. Note also that the *j* might have be assigned to the letter *h* just as well as the letter *u*; such decisions are arbitrary but consistency across words should be enforced. Details of how a set of pronunciations may be determined for a set of words can be found in Section 2.2; for now, we will assume that we have an inventory of such pronunciations in which each pronunciation consists of a string of phones.

We now may write the probability of pronunciation string $\mathcal{R}$ as

$$p(\mathcal{R} \mid \mathcal{L}) = \prod_{i=1}^{n} p(r_i \mid r_1^{i-1}, l_1^n) \quad (2)$$

$$\approx \prod_{i=1}^{n} p(r_i \mid r_{i-5}^{i-1}, l_{i-5}^{i+5}) \quad (3)$$

$$(4)$$

We assume that the probability of the pronunciation r_i only depends on the current letter, the five previous letters, the five following letters, and the five previous pronunciations. We define this to be the *context* of the current pronunciation.

Estimating $p(\mathcal{R} \mid \mathcal{L})$ in a straightforward fashion by counting the number of times r_i occurs in a particular context is an impossible task - there are just too many contexts. Instead, we will map contexts onto a relatively small number of equivalence classes with the aid of a *decision tree*. The decision tree will partition the contexts into classes by asking binary questions about the context elements; e.g., "Is the next letter a vowel?", "Is the previous pronunciation a plosive?", etc. The leaves of the decision tree will represent the equivalence classes. At each leaf is a probability distribution on the allowable pronunciations. Lucassen [4] describes in substantial detail techniques for constructing decsision trees to model $p(\mathcal{R} \mid \mathcal{L})$; we will limit ourselves to briefly outlining the technique and also discuss some modifications to Lucassen's original procedure.

2.1 Decision Tree Construction

Assume we have for each letter a collection of data. Each data item consists of the pronunciation of the letter in a particular context, r, and a set of *context elements*: the five letters preceeding the current letter in this context the five letters following the current letter, and the the five previous pronunciations, We assume we have an inventory of B binary questions (Lucassen [4] describes how such a set of questions may be developed). Each binary question b_j partitions the letters or pronunciations into two subsets. Each b_j may be applied at each context element of each data item. Therefore, for each combination of binary question and context element, the data will be partitioned into two sub-collections; corresponding to each sub-collection is a probability distribution over the pronunciations r. We select the question b_j and context element k that minimizes the average entropy of the two distributions corresponding to the two sub-collections. We now repeat the above procedure for each of the sub-collections individually. The data is thus successively split into smaller and smaller collections until some termination criterion is met; the result is a tree of binary questions about different context elements. The leaves of the tree consist of probability distributions $p(r)$ which can be used in equation 2, above.

There are various termination criteria that can be used to avoid overtraining trees by growing them too deeply. Lucassen used a combination of seven different termination criteria at a node; Breiman [7] recommends growing the tree quite deeply and then pruning it back to minimize the entropy on held-out data. We use the following two techniques, which are quite simple to implement. First, a threshold is placed on

the product of the number of samples at a node and the entropy of $p(r)$. Second, the data is divided into two parts, $\mathcal{A}$ and $\mathcal{B}$. The data in part $\mathcal{A}$ is first used to construct the tree. When a question b_j and context position k is selected using data from part $\mathcal{A}$ at a particular node in the tree, data from part $\mathcal{B}$ is also propagated to this node and the average cross entropy between the $\mathcal{A}$ and $\mathcal{B}$ data is calculated. If the reduction in cross entropy is below a threshold, the node is not expanded, otherwise the $\mathcal{B}$ data is split as the $\mathcal{A}$ data and used at the next level of tree construction. This way, spurious questions that result from a lack of training data may be eliminated. After a tree using $\mathcal{A}$ is constructed, the roles of $\mathcal{A}$ and $\mathcal{B}$ are reversed and a second tree is grown. The predictions from both trees are then averaged to compute $p(\mathcal{R} \mid \mathcal{L})$.

In order to be robust with respect to new data, it is usually important to smooth the distributions at the leaves of the tree. This is typically done via deleted estimation [5]: one tries to maximize the probability of some held-out data with the model by adjusting a set of smoothing coefficients. The most powerful technique is to smooth the distribution at a leaf with a linear combination of the distributions leading up to the root of the tree from the leaf. The main disadvantage of this scheme is that it requires that each leaf be represented in the held-out data. There are various solutions to this problem [4, 6] but they require very careful manipulation of the training and held-out data. Instead, we have developed a simple recursive smoothing scheme that operates in the following fashion. Held-out data is poured down the tree from the root. Each node is assigned a class that is a function of the number of held-out data samples that appear. Each node n in the tree is smoothed according to:

$$p'_n = \lambda_1(c_n)p_n + \lambda_2(c_n)p'_{f(n)} + \lambda_3(c_n)u \qquad (5)$$

where p' denotes the smoothed distribution, p, the unsmoothed distribution on held-out data, $f(n)$, the father node of n, $c(n)$, the count class of node n, and u, the uniform distribution. The λs are computed via deleted estimation [5]; the p's may be computed recursively from the root down for each iteration of deleted estimation. The main disadvantage of this scheme is that it strongly favors node distributions close to the leaf in question for smoothing; for trees with many levels, this may be disadvantageous.

2.2 Data Preparation

The main task in data preparation for decision tree construction lies in determining a series of pronunciations for the letters and in extracting the data items for each letter. It is assumed that there exists a source of words matched against phone strings. Our primary source of such material was an online version of Webster's 7th Collegiate Dictionary. To this we added the phonetic baseforms from our 20K word recognition system, yielding a total of roughly 80K word-phone string pairs. A substantial amount of time was spent "completing" the entries in Webster's - inflections and variant pronunciations were indicated by a shorthand notation that was not trivial to unravel. In addition, the notation the dictionary used to indicate pronunciation was not consistent with the notation in our recognition system; notation in Webster's had to be converted to match our 'style' of writing baseforms. For example, Webster's makes many more distinctions between vowels than we do; certain endings are consistently transcribed differently; e.g., we typically represent -ment as 'm eh0 n t' while in Webster's it is found as 'm uh0 n t', etc.. Converting pronunciation formats to be consistent with local conventions is a generic problem in trying to utilize online dictionaries and often can take more time to solve that to employ the data.

Given such a set of data, the next step is to determine a set of pronunciations for each letter. First, an initial guess is made and a set of pronunciations is selected for each letter. A hidden markov model is generated for each letter whose outputs represent the allowable pronunciations for the letter. An HMM for the letter *d* is shown in Figure 1.

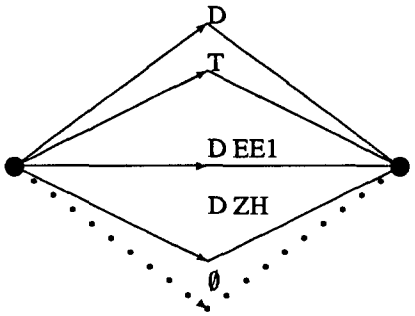

Figure 1: Hidden Markov Model for Pronunciations of the letter *d*.

There are five permissible pronunciations of *d*: *D* as in 'dog', *T* as in 'passed', *D EE1* as in 'PhD', *D ZH* as in 'graduate', and ∅ as in the second 'd' in 'ladder'. The HMM for a word may be constructed by concatenating the HMMs for the letters; the models are now trained using the phone strings as output data. Once the models are trained, a Viterbi alignment is made of all the phone strings against the letter strings for each word. A certain number of the alignments will fail, either because the original pronunciation set was deficient or because there was an error in the original phone string for the word. Based on observing the misalignments, one may augment the original pronunciation set or correct an incorrect baseform, and repeat the procedure as many times as might be necessary. One may then simply obtain the data items for each letter from a combination of the Viterbi alignment and the pronunciation inventory; however, we find that there is often complete arbitrariness the way certain letter strings are aligned against phones; for example, sometimes the phone *T* will be aligned against the first *t* in the letter string "*tt*" and other times, the second. To enforce homogeneity, it is sometimes useful to post-process the Viterbi alignment. The baseforms are written using a set of 50 phones; we typically develop an inventory of 130 pronunciations. Typical letters average 5 pronunciations; some vowels, such as 'u', have up to 13 different pronunciations.

3 Decoding New Baseforms

To determine the best phonetic baseform using the concept of pronunciations, we can rewrite equation 1 in terms of pronunciation strings to find $\mathcal{R}$ such that:

$$\underset{\mathcal{R}}{argmax}\, p(\mathcal{U} \mid \mathcal{R})\, p(\mathcal{R} \mid \mathcal{L}) \tag{6}$$

In the above equation, $p(\mathcal{U} \mid \mathcal{R})$ may be identified with the acoustic model component of a standard Hidden Markov Model based speech recognition system, and $p(\mathcal{R} \mid \mathcal{L})$ with the language model component. The task of determining a phonetic baseform by combining acoustic information with spelling-to-sound rules is substantially simpler than the typical speech recognition problem. First, the 'vocabulary' is not large, typically 130 pronunciations. Second, the number of 'words' in each 'sentence' is determined by the length of the letter string. However, the number of language model states, determined by the number of tree leaves, can be quite large (on the order of several thousand) thus making a Viterbi-type decoder somewhat unwieldy. Instead, a simple stack decoder [5] was implemented to determine the phonetic baseforms. For the n^{th} letter of the word in question, we define the score Δ to be

$$\begin{aligned} \Delta(l_1^n, r_1^n) \;=\; & \alpha \log p(u \mid r_n) + \\ & \log p(r_n \mid r_{n-5}^{n-1}, l_{n-5}^{n+5}) + \\ & \Delta(l_1^{n-1}, r_1^{n-1}) \end{aligned} \tag{7}$$

where subscripts and superscripts refer to strings of letters or pronunciations. The first term on the right just represents the the acoustic model score in a typical HMM-based system for the pronunciation of the n^{th} letter, and the second term can be computed from the spelling-to-sound rules. Note that Δ may be computed recursively by proceeding left to right through the letters of the word. Since each letter in the word can have many different pronunciations, we only examine for the n^{th} letter the pronunciation strings associated with the top m values of Δ from the $(n-1)^{th}$ letter; we have observed that $m = 16$ or $m = 32$ is usually satisfactory. We have also found it necessary to premultiply $\log p(u \mid r_n)$ by a factor α between .1 and .5 to obtain better baseforms. This crudely compensates for poor modelling of time correlation in the Hidden Markov Models.

The above algorithm is only suitable for producing a baseform when there is a single utterance of each word. When there are multiple utterances, we must modify the algorithm slightly. For each utterance of a particular word, we use the above algorithm to produce the top m-scoring pronunciations rather than a single pronunciation, and separately keep track of the score due to the acoustic component and the spelling-to-sound rules. We form the union of all pronunciations across all utterances for each word, and from those pronunciations, find the pronunciation $\mathcal{R}$ to maximize

$$\sum_{i=1}^{p} \beta \log p(\mathcal{U}_i \mid \mathcal{R}) + \log p(\mathcal{R} \mid \mathcal{L}) \tag{8}$$

where $\mathcal{U}_i$ is the i^{th} utterance, and β, a weighting factor similar to α above, is set to approximately .3.

4 Baseform Determination Performance

Our goal was to use the above technique to determine phonetic baseforms for words not in the current speech recognition system's vocabulary of 20000 words. To examine performance, we found the most frequently occurring words not in our 20000 vocabulary from a corpus of one million words of recent internal electronic mail. We selected a random subset of 500 words from this list, and had two talkers each read the list of words on two different days. Each word was read four times each in a single sentence on both days; e.g., "hello hello hello hello". Baseforms were produced using the above techniques under several conditions; in all cases, the baseforms produced were used in conjunction with the 20000 baseforms already in the recognition systems' vocabulary to recognize the second four repetitions recorded from each talker. All recognition took place with a language model that predicted each word uniformly. Talkers recorded our 100 standard training sentences [10] to train the parameters of the Hidden Markov Models. Endpoints of words were located automatically by obtaining a Viterbi alignment of the VQ output against the top baseform produced by the spelling to sound rules. This introduced some errors into the results (see below).

The conditions examined were

1. Spelling-to-sound rules alone used to create baseforms. Weight for acoustic component (α) set to zero.
2. Baseforms made from a single utterance.
3. Baseforms made from all four utterances.
4. Baseforms made from a single utterance; spelling-to sound rules score contribution set to zero.
5. Baseforms made from all four utterances; spelling-to-sound rules score contribution set to zero.

Note that the fourth and fifth conditions are *not* identical to completely ignoring the spelling of the word; the number of letters in the word will determine the length of the pronunciation string (though letters can be given the null pronunciation).

Only 31% of the 500 words were covered by the 80000 words used to determine the spelling-to-sound rules. For that matter, we found that even a much larger word list, generated by combining a list of 40000 common acronyms with the 20000 words in our vocabulary and 250000 words in the Shoup dictionary [11], only covered 60% of these 500 words. Most of the missing words consist of names, acronyms, and specialized jargon, once again illustrating that straightforward dictionary lookup procedures are not adequate for adding words to a vocabulary.

Recognition results for each of the two talkers across all five conditions are shown in Table 2, and compared to results on handwritten baseforms. It can be seen that neither the spelling-to-sound rules nor the acoustics alone seem to be adequate to generate baseforms for recognition, when compared

to handwritten baseforms. Spelling-to-sound rules when combined with four utterances of the word produce baseforms that perform quite competitively with handwritten baseforms; spelling-to-sound rules with a single utterance, or acoustics alone with four utterances yield slightly worse recognition results. Note that the level of recognition performance would be substantially higher in actual recognition applications; natural sentences would be dictated and a strong language model could be used.

Condition	T1	T2
Handwritten baseforms	29%	32%
Spelling-to-sound rules alone	53%	58%
Single utterance	33%	36%
Four utterances	28%	32%
Single utterance, acoustics alone	42%	43%
Four utterances, acoustics alone	33%	36%

Table 2: Recognition performance as a function of baseform construction techniques.

Visual inspection of the baseforms generated by spelling-to-sound rules alone indicated that approximately one-third of the baseforms had errors in them, where errors are defined as substantial discrepancies between the artificially generated baseforms and handwritten ones that had a fair chance of resulting in recognition errors. Visual inspection of the baseforms generated by combining spelling-to-sound rules with four utterances of each word for Talker 1 indicated approximately one-tenth of the baseforms contained some error (as defined above). Approximately one-half the errors occurred on names, one-quarter on acronyms, and one-quarter on actual words. For comparison, the list of 500 words contained 60% words or jargon, 35 % names, and 5% acronyms. An acronym is considered to be not only strings of letters but combinations of strings of letters and words; e.g. 'PCSTORE' is pronounced 'pee-see-store'. One error ('AAA', pronounced 'triple-A') resulted from the word's pronunciation having essentially nothing to do with its spelling. Table 3 lists 10 of the sample errors (chosen at random among the 50).

Some of the above errors, as for "dingles", "Gonzales", "megapels" "Stefanos" and "cepstrum", never resulted in recognition errors, in spite of what subjectively seemed to be substantial errors in the baseforms. The acronym errors were traced a flaw in the endpoint detection algorithm. Endpoints were detected via Viterbi alignment against baseforms produced by the spelling-to-sound rules alone; the baseforms are so terrible for acronyms that substantial misalignment occurred during endpoint detection, generating faulty input to the baseform generator. Very few acronyms were present in the data used to generate the spelling-to-sound rule trees; inclusion of such data might have generated better baseforms. Finally, it should be pointed out that both talkers were not always consistent across recording sessions in their pronunciations of words. For example, the name "Kanevsky", pronounced "kq uh0 n eh1 v s k ee0", was mispronounced in the first recording session as "kq ei1 n v ee1 s k ai1", and the word "cations" was initially pronounced "kq ae1 t ai1 uh0 n z" and then "k ei1 sh uh0 n z". Therefore, even "correct" baseforms never guarantee success in speech recognition!

Word	Baseform	Comment
dingles	d i1 ng g l z	Missing uh0 before "l"
Gonzales	g uh0 n z aa1 l z	Missing eh0 before "z"
Irwin	uh0 er0 w i1 n	Inserted uh0; er0 should be er1
megapels	m eh1 g uh0 p pq uh0 l z	Last uh0 should be be eh1
NBS	eh1 s	Acronym
ODAS		Acronym; no baseform produced
Stefanos	s t tq eh1 f uh0 n au1 ug z	au1 should be ou1
Yong	j aa1 n	n should be ng
BC	b k kq	Acronym
cepstrum	tq sh eh1 p pq s t tq r uh1 m	tq sh should be kq

Table 3: Sample errors in automatic baseform generation.

5 Discussion and Conclusions

Insofar as the automatically generated baseforms produce results at least as good as handwritten ones if the user is willing to say each word multiple times, we view this technique as successful. However, the spelling-to-sound rules obviously have more problems with names and acronyms than actual words; we attribute this at least in part to the fact that there were relatively few names and acronyms in the data used to construct the spelling-to-sound rule trees. We feel that including such information would give the spelling-to-sound rules substantially more power, and are currently working on obtaining data from such sources to improve the quality of baseform construction.

It is hard to compare the performance of this technique with techniques that try to deduce the baseform from the spelling alone. Typical figures quoted are between a 3% and 5% error rate in running text. Crudely speaking, we have observed that a fixed 20000 word vocabulary covers about 93% of completely new text across a variety of sources, implying 7% of the words encountered will be new and must be added to the vocabulary. From the previous section, we calculated a 10% error rate on such new words in constructing baseforms, which would imply we would generate incorrect baseforms for less than 1% of words in running text. It is clear that this hardly impacts the 5% recognition error we obtain in the the laboratory.

In conclusion, we believe we have a viable technique for generating good phonetic baseforms for new words in speech recognition systems. Work is currently progressing to incorporate more data in the above algorithms, and to employ this technique in other components of the speech recognizer to

supply us with the ability to adapt the phonetic baseforms for words already in the vocabulary of the recognizer.

References

[1] L. R. Bahl, P. F. Brown, P. V. deSouza, R. L. Mercer, and M. A. Picheny, "Acoustic Markov models used in the tangora speech recognition system,"

[2] L. R. Bahl, R. Bakis, P. V. deSouza, and R. L. Mercer, "Obtaining candidate words by polling in a large vocabulary speech recognition system," in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, (New York City), pp. 489–492, April 1988.

[3] D. H. Klatt, "Review of text-to-speech conversion for english," *Journal of the Acoustical Society of America*, vol. 82, pp. 737–793, September 1987.

[4] J. M. Lucassen and R. L. Mercer, "An information-theoretic approach to the automatic determination of phonemic baseforms," in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 42.5.1–42.5.4, 1984.

[5] L. R. Bahl, F. Jelinek, and R. L. Mercer, "A maximum likelihood approach to continuous speech recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-5, pp. 179–190, March 1983.

[6] L. R. Bahl, P. F. Brown, P. V. deSouza, and R. L. Mercer, "A tree-based statistical language model for natural language speech recognition," *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol. ASSP-37, pp. 1001–1008, July 1989.

[7] L. Breiman, J. H. Freidman, R. A. Olshen, and C. J. Stone, *Classification and Regression Trees*. California: Wadsworth International Group, 1984.

[8] P. A. Chou, *Applications of Information Theory to Pattern Recognition and the Design of Decision Trees and Trellises*. PhD thesis, Stanford University, June 1988.

[9] A. Averbuch *et al.*, "An IBM-PC based large-vocabulary isolated-utterance speech recgnizer," in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, (Tokyo, Japan), pp. 53–56, April 1986.

[10] A. Averbuch *et al.*, "Experiments with the tangora 20,000 word speech recognizer," in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, (Dallas, Texas), pp. 701–704, April 1987.

[11] J. E. Shoup, "American English orthographic-phonemic dictionary," Air Force Office of Sponsored Research Report AD-763 784, Speech Communications Reserach Laboratory, Inc., 1973.