

MAXIMUM LIKELIHOOD ESTIMATION OF
MULTIPLE PRONUNCIATIONS FOR PROPER NOUNS

By

Neeraj Deshmukh

A Dissertation Proposal
Submitted to the Faculty of
Mississippi State University
in Partial Fulfillment of the Requirements
for the Degree of Doctor of Philosophy
in Electrical Engineering
in the Department of Electrical and Computer Engineering

Mississippi State, Mississippi

June 1999

TABLE OF CONTENTS

LIST OF TABLES	ii
LIST OF FIGURES	iii
CHAPTER	
I. INTRODUCTION	1
II. MAXIMUM LIKELIHOOD MODELING	9
III. REVIEW OF CURRENT TEXT-TO-SPEECH TECHNOLOGY.....	38
IV. PRONUNCIATIONS DATABASE.....	61
V. PROPOSED EXPERIMENTS AND EVALUATION.....	69
VI. REFERENCES	78

LIST OF TABLES

TABLE	Page
1. A list of phoneme symbols used for representing the pronunciations of proper nouns.	63
2. Comparative performance of various name-pronunciation systems on a 400 surname test set (courtesy [49]) compared to performance of test systems on a similar test set of 400 names.	74

LIST OF FIGURES

FIGURE	Page
1. Use of sliding context in creating n-tuples of letters from the spelling of a proper noun, illustrated here for the surname “Amber” with the pronunciation “@ m b &r”. A context length of 0 means each letter is treated individually, a context of 1 indicates each letter considered along with its immediate predecessor and follower, and so on. The symbol “_” indicates a blank (silent) letter or phoneme.....	7
2. Schematic representation of the statistical system to automatically generate multiple likely pronunciations of proper nouns given the spelling.	8
3. The training sequences generated for the word “Matt” with nominal pronunciation “m @ t” using a 5-letter context. The aligned pronunciation is “m @ t_”.....	13
4. The phone space for the vowels for a 18500 surnames pronunciation dictionary. The axis contains indices of all the 44 phonemes that each letter can map to. The axis contains the frequency of letter-phoneme correspondence plotted on a logarithmic scale.	16
5. Topology for a multilayered feedforward stochastic neural network for pronunciation generation. For the example word “Epstein” pronounced as E p s t a I n — the input layer connects to the bit-encoded letter strings of different context	

lengths, the many hidden layers capture the letter-to-sound features in the input bit stream. The output is another bit stream which encodes the phoneme.	19
6. Typical neuron connections in a Boltzmann machine network, along with the Boltzmann distribution function that governs the activation probability of a neuron. The neuron activation function is also illustrated.	22
7. A schematic overview of the simulated annealing backpropagation algorithm used to train the multilayered stochastic neural network.	25
8. Occurrence statistics of phonemes in a proper nouns pronunciation dictionary indicates the variation in the probability of occurrence of phonemes.	33
9. Schematic procedure for generating pronunciation of a word from its spelling using letter-to-phoneme rules and dictionary lookup.	45
10. Schematic procedure for generating pronunciation of a proper noun from its spelling using dictionary lookup, morphology and letter-to-phoneme rules.	48
11. The dynamic alignment procedure illustrated for the word “Wright” and nominal pronunciation “_ 9r aI _ _ t”. The aligned pronunciation is “_ 9r aI _ _ t”.	67

CHAPTER I

INTRODUCTION

*Dearest creature in creation,
Study English pronunciation.
I will teach you in my verse
Sounds like corpse, corps, horse, and worse.
I will keep you, Suzy, busy,
Make your head with heat grow dizzy.
Tear in eye, your dress will tear.
So shall I! Oh hear my prayer.*

*Just compare heart, beard, and heard,
Dies and diet, lord and word,
Sword and sward, retain and Britain.
(Mind the latter, how it's written.)
Now I surely will not plague you
With such words as plaque and ague.
But be careful how you speak:
Say break and steak, but bleak and streak;
Cloven, oven, how and low,
Script, receipt, show, poem, and toe.*

— *An excerpt from "The Chaos" by Dr. G.N. Trenite
(1870-1946), a Dutch observer of English.*

The ability to correctly pronounce names of entities, such as people, places and organizations, is a critical component of effective verbal communication. In many situations, such as looking up information on a person or a place (e.g. airline reservations, directory assistance etc.), it is customary to alternate between written and oral forms of communication. For instance, in telephone directory assistance the customer enunciates the name of a person to be found, and the operator searches for the corresponding spelling in a database. In the process, it becomes necessary to attribute a correspondence between

the two different representations (viz. textual and verbal) of the words. Such a relationship that matches the spelling of a word with its pronunciation is known as a letter-to-sound mapping.

Usually, the pronunciations of a large proportion of the typical words in a language (e.g. common nouns, verbs, adjectives etc.) can be represented using a reasonably limited set of such letter-to-sound rules. When one comes across the spelling of a new word that has not been encountered before, its pronunciation can often be extrapolated by applying the appropriate letter-to-sound rules to various parts of the word [1].

This strategy, unfortunately, is not very successful in case of proper nouns (i.e. names of people, places etc.). For instance, the surname “Brignac” will be pronounced using the standard English letter-to-sound rules to sound like “brig”+”knack”. However, this is actually a surname of French origin and pronounced as “brin”+”yak” (to rhyme with “cognac”).

As illustrated above, proper nouns often have a different morphology and phonology as compared to the typical words, and therefore the standard letter-to-sound mapping is unable to accurately predict a pronunciation for such words [2].

This problem becomes even more pronounced for the text-to-speech or speech-to-text conversions involved in human-machine interactions. An early study conducted in 1985 [3] revealed that a large number of commercial speech synthesizers mispronounced almost 25% of the 2000 most common American surnames. Since then, significant inroads have been made in improving the text-to-speech (TTS) technology. However, the problem of automatically generating acceptable and intelligible

pronunciations for proper nouns still remains unresolved.

Similarly, proper nouns are known to constitute a majority of errors in general-purpose conversational speech recognition tasks [4]. As more and more applications employ voice-driven interfaces, their dependence on accurate recognition of proper nouns also increases and the need for effectively modeling the pronunciations of proper nouns intensifies.

The problem of accurately predicting pronunciations for proper nouns is compounded by the following factors:

1. Many proper nouns (such as surnames) evolve from ethnically heterogeneous sociolinguistic factors that have no commonality with the general pronunciation framework of the English (or any other) language, and are often impossible to model with a reasonably large and systematic rule set.
2. The pronunciations also reflect the differences in the source language and the extent of Anglicization (i.e. assimilation into the English language) [5]. Many times, pronunciation of unfamiliar names uses non-English letter-to-sound correspondences from the native language [6].
3. The notion of a so-called “correct” pronunciation varies from region to region or individual to individual in an essentially idiosyncratic fashion [7].

Consequently, many proper nouns have multiple “correct” pronunciations; and a large percentage of proper nouns have no obvious letter-to-sound mapping rules that can be used to generate these pronunciations. In order to transcend the constraints imposed by these factors, it is necessary for an optimal voice-based interface to automatically generate

a list of plausible pronunciations for a given proper noun.

The spelling of the proper noun is often the only clue available towards its possible pronunciation. Therefore, an approach that statistically relates the letters to corresponding phonemes appears to be reasonable towards developing a letter-to-sound function that is applicable to proper nouns.

Motivation

Since a large proportion of the typical words in a language can be represented in terms of a letter-to-sound rule set, most text-to-speech (TTS) systems use a collection of such rules along with an exceptions dictionary to generate pronunciations. However, the pronunciations of proper nouns do not follow the typical letter-to-sound rules of the language. Therefore, TTS systems employ rule sets specifically crafted for generating proper noun pronunciations. These rule sets need to be quite extensive and the size of the associated exceptions dictionary quite large in order to cover the whole range of names, thus making the systems cumbersome and expensive. Moreover, such rule-based systems are constrained to generate only one pronunciation for a given proper noun, and do not generalize gracefully to names not covered by the rule set.

On the other hand, a statistically generated maximum likelihood model of the functional relationship between proper noun spellings and pronunciations is able to capture the underlying letter-to-sound mapping without taking recourse to hard-coded rules. A system based on such a statistical model can generate the most probable pronunciation of a proper noun that may not be part of the training data for the model.

Moreover, it can produce a list of plausible pronunciations rank-ordered by the likelihood measured against the letter-to-sound statistical model.

The stochastic framework also allows introduction of additional knowledge sources such as language-dependent statistics of the occurrence of certain phonemes, which further improves the likelihood of finding the correct pronunciations for a given proper noun.

Maximum Likelihood Approach

If the spelling of a proper noun is given in terms of a sequence of alphabetical characters

$$I = (i_1, i_2, \dots, i_N) \quad (1)$$

and if pronunciation of proper nouns are denoted by phoneme sequences such as

$$O = (o_1, o_2, \dots, o_M) \quad (2)$$

then the most likely pronunciation O^* for a given spelling I is given by the phoneme sequence with the maximum likelihood given the input letter sequence.

$$O^* = \operatorname{argmax}_O p(O/I) \quad (3)$$

Using Bayes' rule, this can be simplified to include the posterior probability of the spelling given the phoneme sequence $p(I/O)$ and the probability of occurrence of the pronunciation or phoneme sequence $p(O)$ out of all the possible phoneme sequences.

$$O^* = \operatorname{argmax}_O \frac{p(I/O)p(O)}{p(I)} \quad (4)$$

Since the denominator term is common across all possible phoneme sequences, it has no effect on the maximization process and therefore can be ignored from the calculations. As a result, the maximum likelihood representation takes the form

$$O^* = \operatorname{argmax}_O p(I/O)p(O) \quad (5)$$

The posterior probability $p(I/O)$ is a representation of the statistical relationship between the letters and the phonemes (i.e. the letter-to-sound map). The probability of the phoneme sequence $p(O)$ indicates language-related information and provides additional clues as to the possible pronunciations.

Proposed Framework

The problem of generating a list of possible pronunciations given only the spelling of a proper noun can be interpreted as one of mapping each letter in the spelling to a corresponding phonemes. This mapping can be generated statistically as a classification process where each letter is assigned a phoneme identity based on its context in the spelling of the name. To obtain the most probable pronunciations for a given proper noun, it follows that each letter should be mapped to phonemes that maximize the likelihood of the resulting pronunciation phoneme sequences.

As described earlier, proper nouns are derived from a number of languages and ethnic roots. Their pronunciations may contain phonemes that are not present in English, but are unique to the language of origin. Therefore, the phoneme set used to describe these pronunciations in a TTS system needs to provide adequate coverage of the phonemes in a

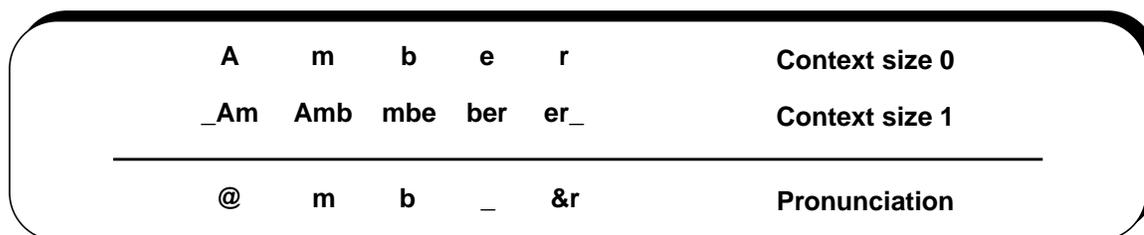


Figure 1. Use of sliding context in creating n-tuples of letters from the spelling of a proper noun, illustrated here for the surname “Amber” with the pronunciation “@ m b &r”. A context length of 0 means each letter is treated individually, a context of 1 indicates each letter considered along with its immediate predecessor and follower, and so on. The symbol “_” indicates a blank (silent) letter or phoneme.

large number of languages. The Worldbet phonetic representation system [8] was developed specifically to achieve this objective, and will be used to transcribe the pronunciations in this work.

In a large number of proper nouns, the pronunciation of each letter is a function of its surrounding letters (e.g. the pronunciation of the letter “a” in “Amber” as opposed to in “Ames”). Therefore, instead of assigning a phoneme to an isolated letter, it is more realistic to associate a phoneme with a letter in context of its neighbors in the spelling. Thus the classification problem transforms into one of attributing phonemes to an n-tuple of letters. Such an n-tuple can be generated by sliding a window of an appropriate context length across the spelling of the name. This approach is illustrated in Figure 1.

A classification model for the letter-phoneme mapping can be constructed in a data-driven fashion, without recourse to any hand-written rules, by training the system with a database of name-pronunciation pairs decomposed into ordered sequences of letter n-tuples and the corresponding phoneme symbols. To predict the pronunciation of a new

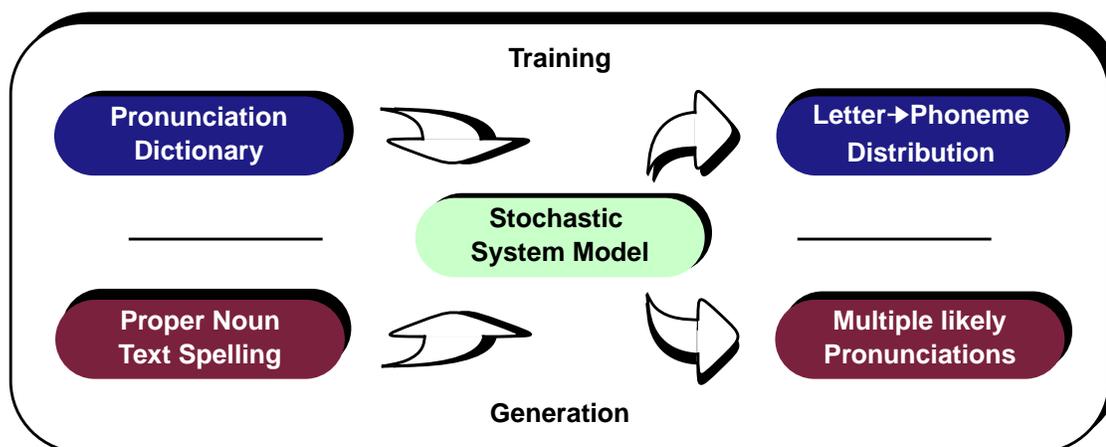


Figure 2. Schematic representation of the statistical system to automatically generate multiple likely pronunciations of proper nouns given the spelling.

proper noun, the trained system will transform its spelling into the appropriately long n-tuples of letters and generate the most likely phonemes for each. This approach is schematically described in Figure 2.

Often, a group of adjacent letters in a proper noun combines to produce a single sound (e.g. “ch” in “Church”, “ough” in “Houghton” etc.) and is therefore associated with a single phoneme. However, each letter (or n-tuple of letters) requires to be associated with a phoneme in the above representation. This problem is handled via the notion of a blank or silent phoneme inserted in the appropriate places in the phoneme sequence. The silent phoneme acts as a placeholder in the model representation, but is never pronounced.

Similarly, for context lengths larger than 0, the first and last letters of the spelling do not have sufficient context before and after, respectively. The required context is obtained by padding the spelling with the appropriate number of blank letters.

The application of both the blank letter as well as the silent phoneme is also illustrated in Figure 1. Both entities are represented by the same symbol “_”.

CHAPTER II

MAXIMUM LIKELIHOOD MODELING

The goal of the proposed research is to develop algorithms that can automatically generate an accurate list of pronunciations for a proper noun using a maximum likelihood estimate of the correspondence between the letters that make up its spelling and the phonemes that constitute its pronunciation.

To that effect, the letter-to-sound relationship can be modeled as a statistical pattern classification paradigm where each letter in the spelling of the proper noun is assigned phoneme symbols that are most likely given the orthographic context of the letter. This is summarized in Equation (5), reproduced here for convenience:

$$O^* = \operatorname{argmax}_O p(I/O)p(O) \quad (6)$$

Here, $I = (i_1, i_2, \dots, i_N)$ is an ordered sequence of N letters that constitute the spelling of the name, and $O = (o_1, o_2, \dots, o_M)$ is an ordered sequence of M phonemes. $p(I/O)$ is a model of the letter-to-phoneme map derived from the statistics of proper noun data and their pronunciations. $p(O)$ indicates properties of the language since it represents the likelihood of the occurrence of the phoneme sequence.

In this framework, scores for various phoneme sequences or pronunciation hypotheses are generated using the letter-to-phoneme posterior likelihoods and the probability of the corresponding partial phoneme sequence. The hypothesis with the maximum overall score is found by performing a dynamic programming search through

all such hypotheses, which yields the best pronunciation for the given spelling.

A Simple Model for Letter-Phoneme Correspondence

Equation (6) can be expanded to include the individual letters and phonemes in the proper noun and its pronunciation respectively as follows —

$$(o_1, o_2, \dots, o_M)^* = \operatorname{argmax} p(i_1, i_2, \dots, i_N / o_1, o_2, \dots, o_M) p(o_1, o_2, \dots, o_M) \quad (7)$$

Assuming statistical independence between adjacent phonemes, i.e. that the sequence of the phoneme symbols represents independent outcomes of a random process, statistically the occurrence of one phoneme in the pronunciation of a name does not influence the chances of appearance of other phonemes.

$$p(o_1, o_2, \dots, o_M) = \prod_{k=1}^M p(o_k) \quad (8)$$

Now, the estimation of the letter-to-phoneme correspondence can be further simplified to a mapping that individually maximizes the likelihood of each phoneme for the given letter sequence.

$$o_k^* = \operatorname{argmax}_{\{o\}} p(i_1, i_2, \dots, i_N / o_k) p(o_k) \quad \dots \forall k \in \{1, 2, \dots, M\} \quad (9)$$

Here, $\{o\}$ is the set of all phonemes. The optimal pronunciation can be constructed by simply concatenating individually most probable phonemes given the spelling as

$$O^* = (o_1, o_2, \dots, o_M)^* = (o_1^*, o_2^*, \dots, o_M^*) \quad (10)$$

and the total probability of the phoneme sequence constituting the pronunciation is given

by simply multiplying the likelihoods of the individual phonemes.

$$p(O^*) = p(o_1^*, o_2^*, \dots, o_M^*) = \prod_{k=1}^M p(o_k^*) \quad (11)$$

By forcing the constraint that each letter in the spelling has a corresponding phoneme associated with it (in case of a cluster of two or more adjacent letters mapping to the same sound, this is achieved by assigning the phoneme to one of the letters and using the silent phoneme as the place holder for the rest — see Figure 1), $M = N$ and Equation (9) further simplifies to

$$o_k^* = \underset{\{o\}}{\operatorname{argmax}} p(i_1, i_2, \dots, i_N/o_k) p(o_k) \quad \dots \forall k \in \{1, 2, \dots, N\} \quad (12)$$

If occurrence of adjacent letters in the spelling is also assumed to be statistically independent, then each phoneme can be associated with a single letter in the spelling. The equation for maximization of the likelihood of the phoneme can now be written for each single letter-phoneme pair for the proper noun.

$$o_k^* = \underset{\{o\}}{\operatorname{argmax}} p(i_k/o_k) p(o_k) \quad \dots \forall k \in \{1, 2, \dots, N\} \quad (13)$$

While Equation (13) presents a fairly simplified representation of the maximum likelihood paradigm, the implicit assumption that the pronunciation of a letter is independent of its surrounding letters is not realistic. As illustrated earlier, it is often the orthographic context of a letter that determines its pronunciation. This dependence of the phoneme associated with a letter on the letter n-tuple centered around this letter can be statistically modeled as a bidirectional or non-causal Markov process.

For a standard n^{th} order Markov process \mathcal{X} , the likelihood of an event given its full history is a function of only the preceding n events. For a sequence of observations x_1, x_2, \dots, x_N , the likelihood of each observation x_k is given by

$$p(x_k/x_1, x_2, \dots, x_{k-1}) = p(x_k/x_{k-n}, x_{k-n+1}, \dots, x_{k-1}) \quad (14)$$

An n^{th} order bidirectional or non-causal Markov process can be thought of as a combination of two n^{th} order Markov processes operating in opposite directions. In other words, the likelihood of an event is a function of the preceding n events, as well as the following n events.

$$p(x_k/x_1, \dots, x_{k-1}, x_{k+1}, \dots, x_N) = p(x_k/x_{k-n}, \dots, x_{k-1}, x_{k+1}, \dots, x_{k+n}) \quad (15)$$

Assuming that the probability of a phoneme o_k corresponding to a letter i_k is a function of a context length n , the likelihood maximization equation takes a form similar to Equation (15).

$$o_k^* = \underset{\{o\}}{\operatorname{argmax}} p(i_{k-n}, \dots, i_{k-1}, i_k, i_{k+1}, \dots, i_{k+n}/o_k)p(o_k) \quad (16)$$

... $\forall k \in \{1, 2, \dots, N\}$

The non-causal estimation of the letters in the spelling follows from the fact that while the phonemes are predicted in a time-sequential manner for each letter, the entire spelling of the name is available in advance to generate letter sequences of any desired context duration.

Equation (16) forms the basis of the sliding-window approach to train the phoneme classification mechanism proposed in this research. Using a sliding window of a

letter					phoneme
_	_	m	a	t	m
_	m	a	t	t	@
m	a	t	t	_	t
a	t	t	_	_	_

Figure 3. The training sequences generated for the word “Matt” with nominal pronunciation “*m @ t*” using a 5-letter context. The aligned pronunciation is “*m @ t _*”.

fixed context length, n -tuple of letters of the proper noun spelling are created with a corresponding phoneme from the pronunciation associated with it. Each sequence can thus be treated as an individual training sample. Figure 3 illustrates how using a context length of $n = 2$ i.e. a window length of $2n + 1 = 5$, a pronunciation is first aligned with the spelling string and then used to generate training sequences. The system can be thus allowed to learn the statistical relationship between each n -tuple of letters and its corresponding phonemes.

Effect of Phonemic Context

In the discussion in the previous section, it was assumed that the consecutive phonemes in the pronunciation are statistically independent of each other (Equation (8)). However, a study of human articulatory patterns reveals that speech sounds i.e. phones are dependent on context [9], and therefore it may be beneficial to assume some statistical dependence between a phoneme in the pronunciation string and its predecessors. This may be modeled as a Markov process. Using a single-phoneme history i.e. a first order Markov

assumption, Equation (8) can now be written as

$$p(o_1, o_2, \dots, o_M) = \prod_{k=1}^M p(o_k/o_{k-1}) \quad (17)$$

As a result, the probability of the phoneme sequence representing a pronunciation of the input proper noun can be estimated as

$$o_k^* = \underset{\{o\}}{\operatorname{argmax}} p(i_{k-n}, \dots, i_{k-1}, i_k, i_{k+1}, \dots, i_{k+n}/o_k) p(o_k/o_{k-1}) \quad (18)$$

... $\forall k \in \{1, 2, \dots, N\}$

A system that uses the phoneme history in addition to the letter context information to generate the pronunciation string will thus be required to evaluate a score for each predicted phoneme based on the estimates of the observation probabilities (i.e. the likelihood of the predicted phoneme given the letter context sequence and the history phoneme), and a search strategy that hypothesizes all the possible phoneme sequences and generates a list of those that have the highest cumulative likelihood scores. This is analogous to the N-best Viterbi decoding used commonly in speech recognition systems to determine the best sequence of phones based on acoustic scores [10, 11, 12].

In general, an m^{th} order Markov process can be used to model the phonemic history, in which case Equation (17) can be re-written as

$$p(o_1, o_2, \dots, o_M) = \prod_{k=1}^M p(o_k/o_{k-1}, o_{k-2}, \dots, o_{k-m}) \quad (19)$$

This is similar to the n-gram language modeling techniques used in speech recognition, and corresponding methods for likelihood determination that include back-off estimates

and smoothing can also be applied to calculate the probabilities of the phoneme sequences.

Estimation of Letter-to-Phoneme Posterior Probability

The expression for maximizing the estimated likelihood of a pronunciation phoneme sequence, as derived in the previous section, is given by

$$p(O^*) = \prod_{k=1}^N p(o_k^*) \quad (20)$$

where

$$o_k^* = \underset{\{o\}}{\operatorname{argmax}} p(i_{k-n}, \dots, i_{k-1}, i_k, i_{k+1}, \dots, i_{k+n}/o_k) \times p(o_k/o_{k-1}, o_{k-2}, \dots, o_{k-m}) \quad (21)$$

for a context length of n for the letters and an m^{th} order Markov assumption for the phoneme sequence.

Thus the likelihood estimation can be split into two distinct operations — one for the *a posteriori* likelihood of the letter sequence given the phoneme (i.e. the letter-to-phoneme correspondence map)

$$p(i_{k-n}, \dots, i_{k-1}, i_k, i_{k+1}, \dots, i_{k+n}/o_k) \quad (22)$$

and the other for the likelihood of the phoneme given its history in terms of previously occurred phonemes in the pronunciation, if any

$$p(o_k/o_{k-1}, o_{k-2}, \dots, o_{k-m}) \quad (23)$$

Estimation of the posterior likelihood as defined by Equation (22) can be treated as

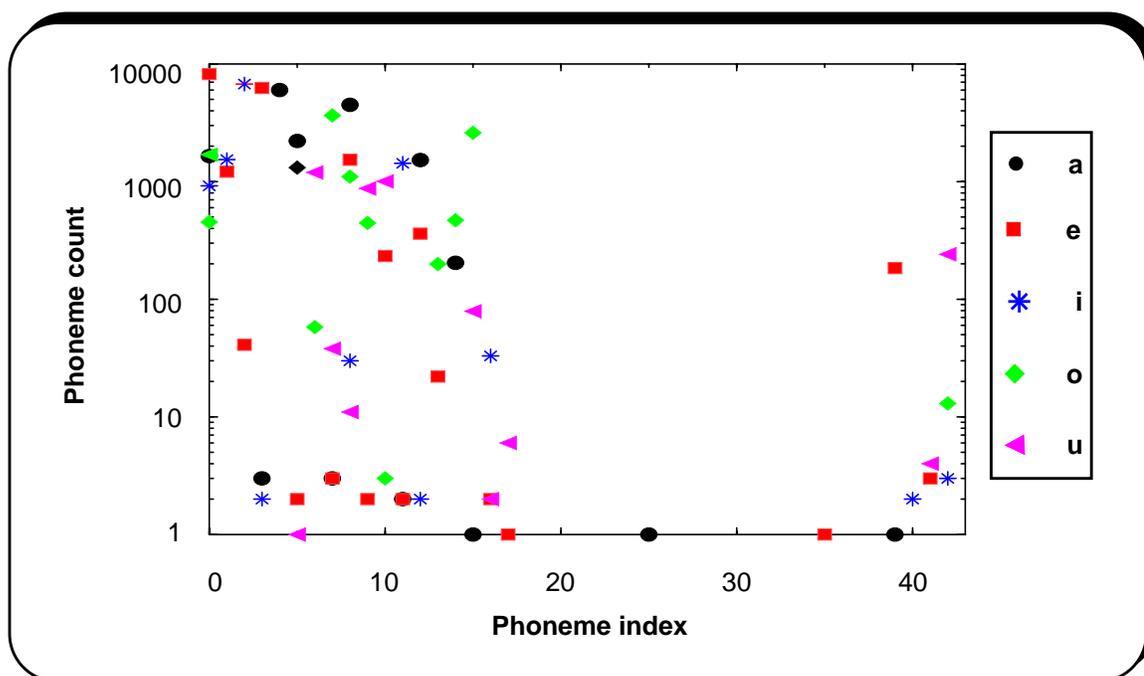


Figure 4. The phone space for the vowels for a 18500 surnames pronunciation dictionary. The x axis contains indices of all the 44 phonemes that each letter can map to. The y axis contains the frequency of letter-phoneme correspondence plotted on a logarithmic scale.

a classification problem, where the classifier is trained to represent the statistics of the association between letter n -tuples and the associated phoneme.

Since the pronunciation of a proper noun is not only a function of the letter n -tuples in its spelling, but also of several non-analytic factors such as ethnicity and dialect, a single letter n -tuple is often pronounced in different ways and therefore gets mapped to multiple phonemes. Consequently, there is frequent overlap among the classification regions for the various phonemes. This problem is illustrated for the vowel space (i.e. pronunciations of only the vowels “a”, “e”, “i”, “o” and “u” as they appear in a pronunciation dictionary of 18500 American surnames) in Figure 4.

Accounting for phonemes corresponding to the other letters in addition to the vowels, it is reasonable to extrapolate that the classification space is highly confusable and nonlinear. Standard linear classification techniques such as minimum mean squared error and nearest neighbor clustering will not be able to satisfactorily assign phonemes to the corresponding letters. Moreover, the statistical model of letter-to-phoneme correspondence is supposed to estimate a probability function for the occurrence of each letter-phoneme pair, a task not achievable with simple classification techniques. Therefore, techniques that can optimally maximize the estimated likelihood of the letter-phoneme pairs are needed to model the statistics of the classification space. Commonly used techniques for such stochastic pattern classification problems are statistical decision trees, artificial neural networks, Markov and hidden Markov models etc.

Statistical decision trees divide the classification space into piecewise linear regions based on intelligent questions about the properties of the input data [13, 14] and assign probabilities that a class corresponds to the input pattern accordingly. A number of existing text-to-speech systems involve decision tree techniques to convert text to phonemes, and the effect of decision trees for generating pronunciations of surnames has been studied in detail in [15]. The performance of such systems suffers from the inability to generalize to the pronunciations of previously unseen names.

Neural networks, on the other hand, use interconnected sets of hidden units where the connection weights are adjusted to model the complex relationships between each class and every input data token [16, 17]. Thus the output of the network is able to capture the inherent functionality of the input data without any *a priori* statistical characterization

or parameterization. The network can reduce large input vectors into small output feature vectors that effectively indicate the classes represented by the input patterns. Often, such features constitute the internal activation patterns of a network rather than the output [18, 19].

Another approach that will be explored in this research is inspired from the methodologies used in solving problems pertaining to automatic recognition of speech, where the system generates a maximum likelihood sequence of speech units (phones, words etc.) using a dynamic programming search through the possible hypotheses. The probabilities are estimated using statistical optimization methods such as the EM (expectation-maximization) algorithm [20].

Likelihood Estimation Using Statistical Neural Networks

Artificial neural networks (ANNs) are an attractive approach for nonlinear classification tasks due to their ability to perform complex computing or classification operations through massive integration of individual computing units, each of which performs an elementary computation.

ANNs are connectionist systems in which the knowledge about the data is distributed over multiple processing units and the net exchange of information between these units determines the behavior of the system. Such networks can be created automatically through incremental learning i.e. by repetitive training on example cases. Multilayered neural networks, in which the internal or hidden units can act as feature detectors that perform a mapping between the input and the output are a class of models

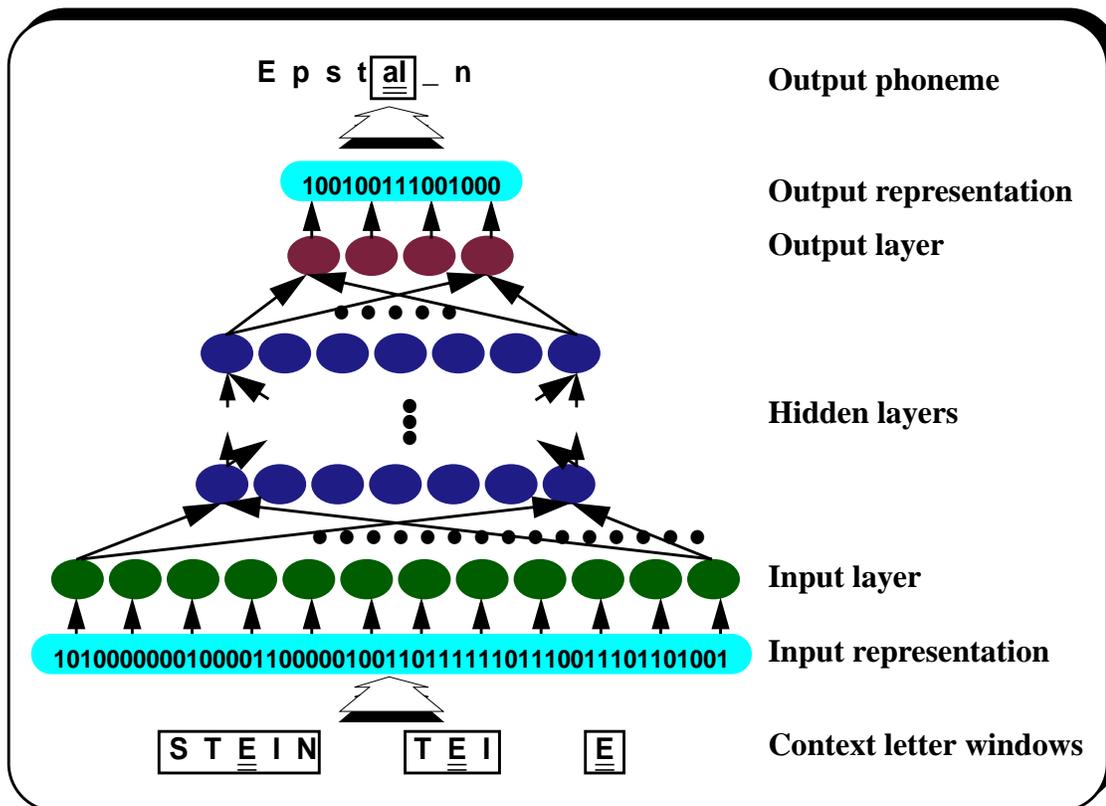


Figure 5. Topology for a multilayered feedforward stochastic neural network for pronunciation generation. For the example word “Epstein” pronounced as *E p s t a l n* — the input layer connects to the bit-encoded letter strings of different context lengths, the many hidden layers capture the letter-to-sound features in the input bit stream. The output is another bit stream which encodes the phoneme.

ideally suited for such applications of letter-to-phone conversion.

A hybrid stochastic neural network [21] is proposed here that combines the principles of multilayered feedforward networks [22, 23] and Boltzmann machines [24]. This network looks at each character in the spelling of the noun in the context of its left and right neighbors and maps such letter substrings to the corresponding sounds. A shift register structure is used to buffer characters as they are input to the system one at a time. Local as well as long-distance constraints can be incorporated in the input data [25] using

shift registers of different context sizes simultaneously. This approach is similar to other time-delay techniques that have become popular in speech recognition systems [26, 27].

The network architecture for the problem of name pronunciation generation is depicted in Figure 5. It is based on the following two design criteria —

- Generally, a relatively small amount of contextual information is sufficient to narrow the range of possible sound correspondences to a small set of likely phonemes.
- Choosing a correct sound from this set may require information occurring at more remote points in the name (such as the identification of spelling patterns unique to the language from which the name is derived).

The network architecture consists of three principal components described as follows —

- An input layer that buffers n -tuples of input letters and maps them to binary-valued inputs — the input character set consists of the 26 letters of the alphabet, plus the blank letter “_” and a few other special characters such as the apostrophe and the period.
- A set of hidden layers that maps such bit-streams into a set of internal states — that derive and store the context-sensitive information regarding the “sounds” such n -tuples produce, and transform the bit-string output of the input layer into some representation of sounds or features corresponding to the pronunciation of the name. The connection weights between the input buffers and the hidden layer are used to represent knowledge about the n -tuple letter sequences.

- An output layer that integrates the long-term and short-term constraints to interpret the groups of letters into a phonetic representation — an indexing system is used to encode the output symbols as phonemes in order to reduce the complexity of the system.

By repeatedly applying the same name as input to the system, different phonetic feature sequences can be produced, corresponding to alternate plausible pronunciations of the name. Thus, the network succeeds in determining not only a nominal or “correct” pronunciation, but also describing all likely pronunciations of the name. The activation probability of each unit in the network also provides a likelihood measure or “score” for each pronunciation.

Standard algorithms used for training multilayered perceptrons, such as simulated annealing [28] and error backpropagation were found to have divergence problems in training the letter-to-phoneme classifier due to the nature of the classification space [29]. Therefore, a modified training algorithm is proposed here for this neural network system.

This algorithm is a variation on the standard simulated annealing gradient descent method. It tries to minimize the asymmetric divergence (an information-theoretic measure of the distance between two probability distributions) between the network energy distributions generated by the reference pronunciation and the hypothesis phoneme string.

Derivation of The Weight-Update Rules

The typical connections between neurons in two connected layers are illustrated in Figure 6. Let $\{w_{ij}\}$ be the set of weights connecting units in the two layers, such that

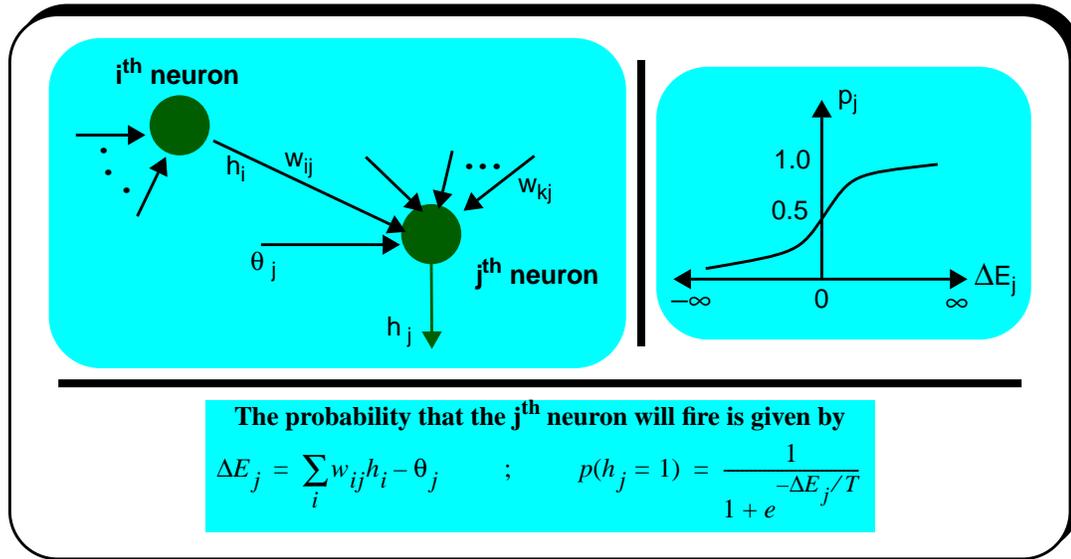


Figure 6. Typical neuron connections in a Boltzmann machine network, along with the Boltzmann distribution function that governs the activation probability of a neuron. The neuron activation function is also illustrated.

$\{h_i\}$ represents the set of output bits of one layer (say the input layer) and $\{h_j\}$ those of the other (output layer). Let α be a global state of this Boltzmann machine neural network corresponding to the case where the outputs represent the hypothesis pronunciation i.e. are set according to the input bits and not clamped externally; and the network is in thermal equilibrium. When the output bits are clamped to their desired values, let the state of the network at thermal equilibrium be represented by $\hat{\alpha}$. If P_α indicates the probability of the system being in the equilibrium state α , then the asymmetric divergence between the distributions at the hypothesis state α and the desired or ideal state $\hat{\alpha}$ is defined as

$$\Psi = \sum_{\hat{\alpha}} P_{\hat{\alpha}} \log \frac{P_{\hat{\alpha}}}{P_\alpha} \quad (24)$$

Assuming that the threshold values at each unit are included in the summation, the total global energy of this system in state α is

$$E_\alpha = \sum_j \sum_i w_{ij} h_i^\alpha h_j^\alpha \quad (25)$$

Therefore differentiating $e^{-E_\alpha/T}$ with respect to a connection weight w_{ij} yields

$$\frac{\partial e^{-E_\alpha/T}}{\partial w_{ij}} = \frac{1}{T} e^{-E_\alpha/T} h_i^\alpha h_j^\alpha \quad (26)$$

The probability that the system under thermal equilibrium conditions ends up in the global state α is given by the following equation [28] —

$$P_\alpha = \frac{e^{-E_\alpha/T}}{\sum_\lambda e^{-E_\lambda/T}} \quad (27)$$

where λ is any global state of the network. Differentiating Equation (27) and substituting Equation (26) the following equation is obtained —

$$\frac{\partial P_\alpha}{\partial w_{ij}} = \frac{\frac{e^{-E_\alpha/T}}{T}}{\sum_\lambda e^{-E_\lambda/T}} \left(h_i^\alpha h_j^\alpha - \frac{\sum_\lambda h_i^\lambda h_j^\lambda e^{-E_\lambda/T}}{\sum_\lambda e^{-E_\lambda/T}} \right) \quad (28)$$

which simplifies to

$$\frac{\partial P_\alpha}{\partial w_{ij}} = \frac{P_\alpha}{T} \left(h_i^\alpha h_j^\alpha - \frac{\sum_\lambda h_i^\lambda h_j^\lambda e^{-E_\lambda/T}}{\sum_\lambda e^{-E_\lambda/T}} \right) \quad (29)$$

This relation is used to compute the gradient of the error function. The derivative of Equation (24) results in

$$\frac{\partial \Psi}{\partial w_{ij}} = -\sum_{\alpha} \frac{P_{\widehat{\alpha}}}{P_{\alpha}} \frac{\partial P_{\alpha}}{\partial w_{ij}} \quad (30)$$

A substitution from yields

$$\frac{\partial \Psi}{\partial w_{ij}} = -\frac{1}{T} \sum_{\alpha} \frac{P_{\widehat{\alpha}}}{P_{\alpha}} P_{\alpha} \left(h_i^{\alpha} h_j^{\alpha} - \frac{\sum_{\lambda} h_i^{\lambda} h_j^{\lambda} e^{-E_{\lambda}/T}}{\sum_{\lambda} e^{-E_{\lambda}/T}} \right) \quad (31)$$

Noting that the sum of probability values over the entire domain is 1 and that the input bits are the same in both the reference and hypothesis cases —

$$\sum_{\alpha} P_{\widehat{\alpha}} = 1 \quad \text{and} \quad h_i^{\alpha} = h_i^{\widehat{\alpha}} \quad (32)$$

if the error in the output bit is termed as $\delta_j^{\alpha} = h_j^{\widehat{\alpha}} - h_j^{\alpha}$, then on further simplification of Equation (31) the relationship between the gradient of the error function and the bit error becomes apparent.

$$\frac{\partial \Psi}{\partial w_{ij}} = -\frac{1}{T} (h_j^{\alpha} - h_j^{\widehat{\alpha}}) h_i^{\alpha} = \frac{1}{T} (h_j^{\widehat{\alpha}} - h_j^{\alpha}) h_i^{\alpha} = \frac{1}{T} \delta_j^{\alpha} h_i^{\alpha} \quad (33)$$

Thus to minimize Ψ it is sufficient to change each weight by an amount proportional to the difference between the expected output and the desired output.

$$\Delta w_{ij} = \eta \delta_j^{\alpha} h_i^{\alpha} \quad (34)$$

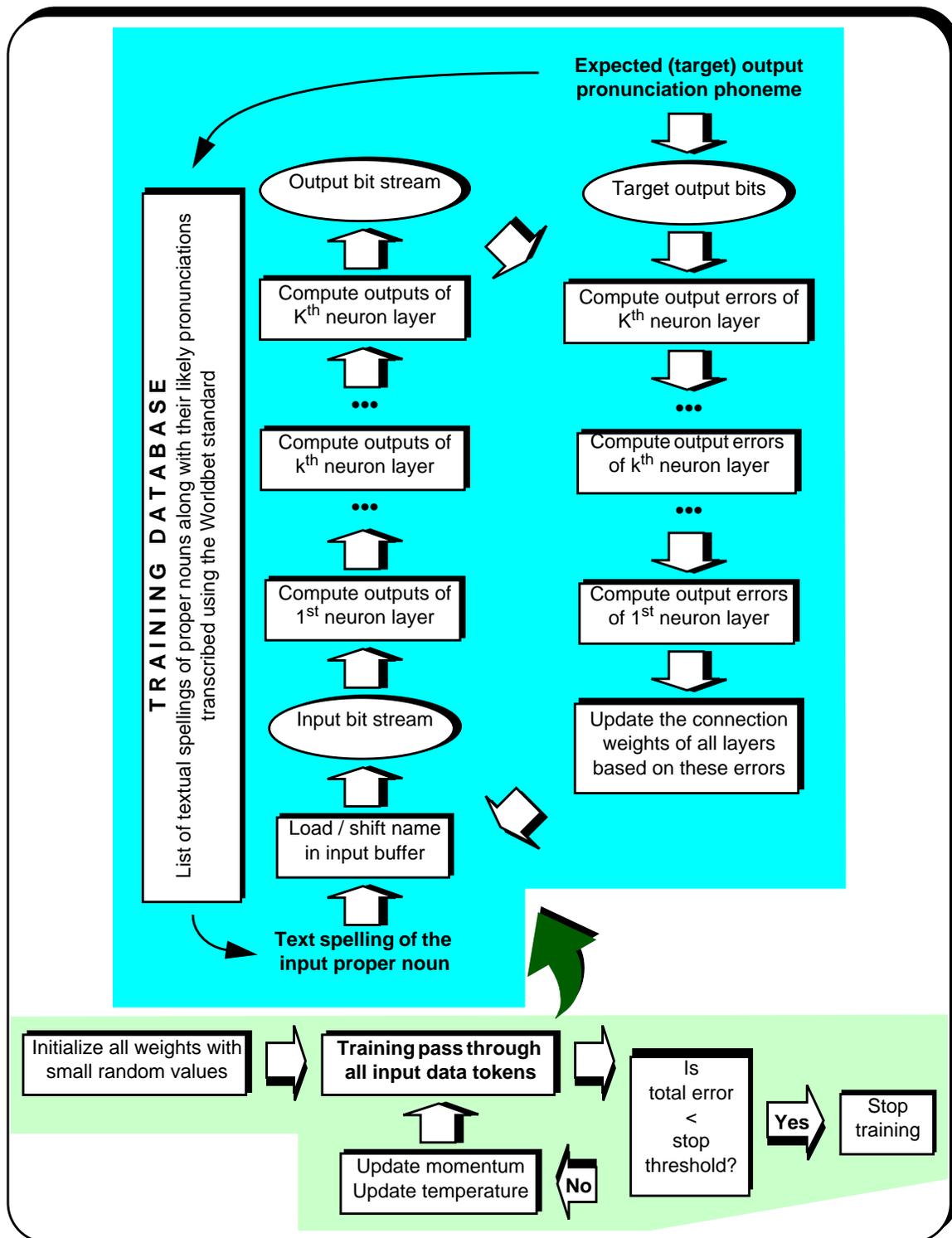


Figure 7. A schematic overview of the simulated annealing backpropagation algorithm used to train the multilayered stochastic neural network.

Here η is a scaling factor that determines the size of each weight change, and in the context of neural network training is called the *learning rate*.

There are a number of possible modifications to the algorithm derived above that affect the learning in terms of speed of convergence. By keeping the scaling factor fairly small the noise in incrementing the weights can be minimized, but a very small value of η also results in a slow learning rate. This can be partly compensated for by adding some momentum to the training. This involves providing some feedback to the weight updates based on the updates for the previous input-output case. The feedback factor μ is called the momentum coefficient, and can be varied to control the direction of learning to some extent. Now the weight update equation takes the form

$$\Delta w_{ij} = \eta \delta_j^\alpha h_i^\alpha + \mu \Delta w_{ij} \quad (35)$$

which is used in training the pronunciation generating system.

The training algorithm is described in detail next. Figure 7 contains a schematic outline of the training process.

Training Algorithm for Stochastic Multilayered Neural Network

Given a set of input spellings along with the corresponding phonetic transcriptions, to compute the set of weights for a stochastically activated network with K number of layers that maps the inputs onto the corresponding outputs the following procedure is used:

1. As there are K layers in the network, layer 1 corresponds to the one clamped

with the inputs and layer K corresponds to the neuron layers that constitute the system output. Let N_k be the number of neurons in the k^{th} layer. Also, let N_0 be the number of bits that are input the first layer of neurons. These bits are the accumulation of the bit-strings corresponding to all the symbols loaded in the input buffer, and hence N_0 is fixed once the context size is decided. Let the input bits be denoted by x_i , the activation levels of the neurons in the k^{th} hidden layer be denoted as h_{k_i} and the output bits of the K^{th} (output) layer be o_i . We indicate the weight connecting the i^{th} neuron in the $k-1^{\text{th}}$ layer to the j^{th} neuron in the k^{th} layer by $w_{k_{ij}}$. t is the index of the number of training loops. $T(t)$ is the *system temperature* in the t^{th} iteration through the training data. Let η be the *learning rate* and $\alpha(t)$ be the *feedback coefficient* or the *momentum* term used to update the connection weights. The learning rate is a fixed constant that characterizes the impact of the output error of a neuron on the weights connected to it. The momentum determines how much the previous training affects the weight update. For $t = 0$, the weights are initialized to small random values (say between -0.1 and 0.1). The respective initial values of the momentum $\alpha(0) = 0$ and the temperature $T(0) = T_0$ are also set. The initial temperature T_0 is a parameter specified by the user.

2. The following steps 3 through 10 are carried out for each iteration $t = 0, 1, 2, \dots$

over all the training vectors.

3. For an input vector $\{x_i\}$, the probability of getting a high output for a hidden layer neuron clamped to it is calculated in terms of its energy gap. The outputs are set to a high or low value using a random number generator that follows this distribution.

$$\Delta E_{1j} = - \sum_{i=0}^{N_0} w_{1ij} x_i \quad (36)$$

$$p(h_{1j} = 1) = \frac{1}{1 + e^{\Delta E_{1j}/T(t)}} \quad (37)$$

4. The output of the units in the first hidden layer is propagated through the network to compute the outputs of neurons in the subsequent layers. Thus for all $k = 2, 3, \dots, (K-1)$ —

$$\Delta E_{kj} = - \sum_{i=0}^{N_{k-1}} w_{kij} h_{k-1i} \quad (38)$$

$$p(h_{kj} = 1) = \frac{1}{1 + e^{\Delta E_{kj}/T(t)}} \quad (39)$$

5. Finally, the output bits of the outermost layer are computed.

$$\Delta E_{Kj} = - \sum_{i=0}^{N_{K-1}} w_{Kij} h_{K-1i} \quad (40)$$

$$p(o_j = 1) = \frac{1}{1 + e^{\Delta E_{Kj}/T(t)}} \quad (41)$$

6. The output bit-string is compared to the bit-string $\{y_i\}$ that corresponds to the expected or target output phoneme. The error in the system output is computed based on the actual output and the target output. Since this error corresponds to the outermost layer, the error for the j^{th} neuron in this layer is denoted as δ_{K_j} .

$$\delta_{K_j} = o_j(1 - o_j)(y_j - o_j) \quad (42)$$

7. The error in the output of a neuron in an earlier layer is computed. The error at the k^{th} layer is calculated by backpropagating the error in the $k + 1^{\text{th}}$ layer and is denoted by δ_{k_j} . For all $k = K - 1, K - 2, \dots, 1$ —

$$\delta_{k_j} = h_{k_j}(1 - h_{k_j}) \sum_{i=0}^{N_k} \delta_{k+1_i} w_{k+1_{ij}} \quad (43)$$

8. The weights are updated using these error values with some feedback from the updates in the previous training pass (see Appendix A for derivation). This feedback is controlled using the *learning rate* η and the *momentum* or feedback coefficient α . For all $k = K, K - 1, \dots, 1$ —

$$\begin{aligned} \Delta w_{k_{ij}} &= \eta \delta_{k_j} h_{k-1_j} + \alpha(t) \Delta w_{k_{ij}} \\ \Delta w_{1_{ij}} &= \eta \delta_{1_j} x_j + \alpha(t) \Delta w_{1_{ij}} \end{aligned} \quad (44)$$

9. Steps 3 through 8 are repeated for the next input token. This is continued till all input tokens are exhausted. A complete training pass through all the input tokens is called an iteration or an *epoch*.
10. The momentum and temperature parameters are updated for the next iteration

through the training data. The momentum term is slowly increased to be small in the beginning and to approach unity as the network runs through more epochs. The temperature is gradually decreased i.e. the system is allowed to cool down as per the simulated annealing paradigm. A common cooling schedule follows an exponential function with a cooling exponent β specified by the user. Therefore —

$$\alpha(t) = 1 - e^{-\beta t} \quad (45)$$

$$T(t) = T_0 e^{-\beta t} \quad (46)$$

11. The network continues to make passes of the training data till the cumulative mean squared error in the output values drops below a suitable threshold. At this juncture the system is said to have achieved convergence.

The training may be stopped according to several other criteria as well. These may include stopping the training once some minimum value of the system temperature is reached, or when the largest increment in any of the connection weights is smaller than a threshold value etc.

Likelihood Estimation Using Context Interpolation

The neural network model attempts to capture the letter-to-phoneme statistical distribution in a non-parametric framework i.e. it poses no restrictions on the form of the distribution. Therefore, a continuous density probability function is estimated. An alternative to training the neural network models is to estimate the letter-to-phoneme

correspondence as a discrete probability distribution.

This is achieved by using ratios of the frequency of occurrence of each letter n-tuple and phoneme pair for a given context length. A back-off count is also maintained for each central letter using an n-tuple of a smaller context length, and the corresponding back-off probabilities estimated. When generating the pronunciation of a proper noun, the most likely phonemes given the letter n-tuple string are output. If a particular n-tuple does not exist in the model, the phoneme likelihood associated with it is calculated by backing off to the next smallest context using the back-off weights.

First, all the distinct phonemes in the pronunciation database are indexed to integers in the range 1 through V , where V is the size of the phoneme set. For a context length of n , the size of the n-tuple is $2n + 1$. If a letter sequence centered around the letter ξ be denoted by ξ_{2n+1} , then let the number of times the phoneme with the index x is associated with this n-tuple be $v(\xi_{2n+1}/x)$. The probability of this letter n-tuple to be associated with this phoneme is then given by

$$p(\xi_{2n+1}/x) = \frac{v(\xi_{2n+1}/x)}{\sum_{y=1} v(\xi_{2n+1}/y)} \quad (47)$$

At the same time, the back-off context is of size $b = n - 1$, with the corresponding letter sequence of the size $2b + 1 = 2n - 1$. The back-off probability of this shorter letter sequence ξ_{2n-1} , which is obtained by stripping the extreme letters of the full-length sequence ξ_{2n+1} , to be associated with the phoneme index x is then given

by

$$\begin{aligned}
 p(\xi_{2n-1}/x) &= \frac{v(\xi_{2n-1}/x)}{\sum_{y=1} v(\xi_{2n-1}/y)} \\
 &= \sum_A p(\xi_{2n+1}/x) \quad \dots A = \{\xi_{2n+1} | * \xi_{2n-1} * = \xi_{2n+1}\}
 \end{aligned} \tag{48}$$

Here, $*\xi_{2n-1}*$ indicates the letter sequence of length $2n+1$ which has any possible letter at each of the two extreme positions, but the central substring is ξ_{2n-1} .

These back-off likelihoods are estimated for $b = n-1, n-2, \dots, 1$. During evaluation, the likelihood of an input n -tuple of letters centered around ξ for a phoneme x is given by

$$p(\xi/x) = \begin{cases} p(\xi_{2n+1}/x) & \dots \text{if exists} \\ p(\xi_{2n-1}/x) & \dots \text{otherwise} \end{cases} \tag{49}$$

The system backs off to progressively smaller context length substrings if the longer letter substring is not found to have a satisfactory count of occurrences in the training data.

Estimation of Phoneme Probabilities

The posterior probabilities of the letter-phoneme pairs can be used directly, without any additional knowledge to generate pronunciations of proper nouns. However, the performance of such a system based only on classification of phonemes is poor [30]. This system essentially assumes a uniform distribution on the occurrence of all phonemes.

However, not all phonemes are equiprobable, as described in Figure 8 by the plot

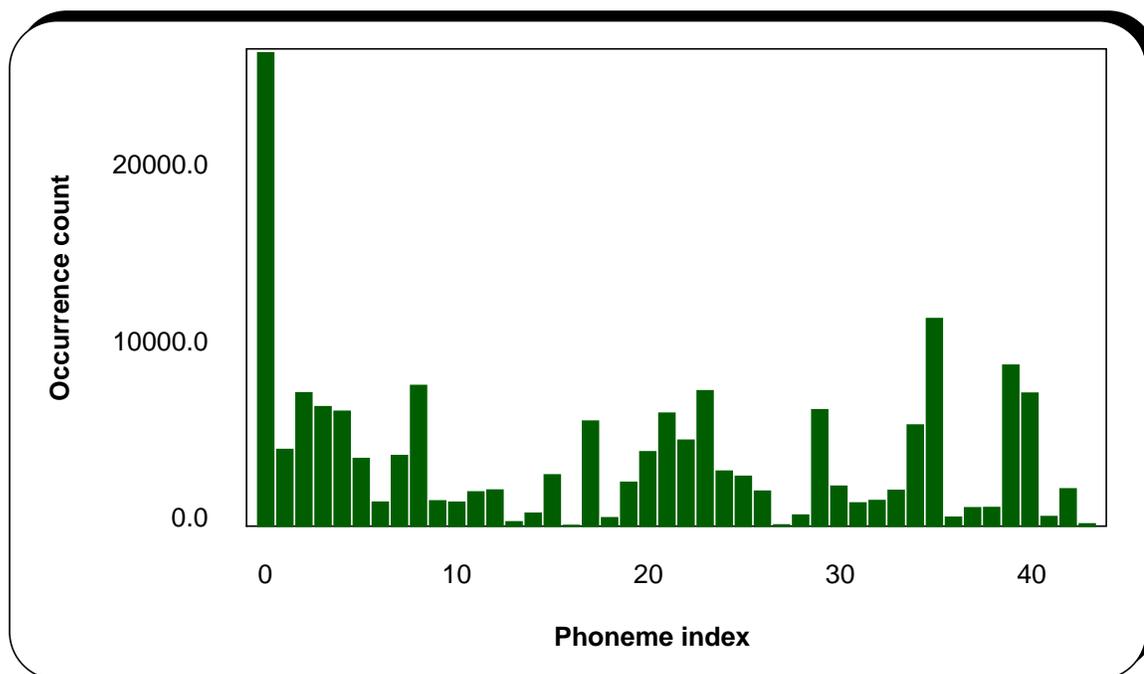


Figure 8. Occurrence statistics of phonemes in a proper nouns pronunciation dictionary indicates the variation in the probability of occurrence of phonemes.

of the occurrence frequencies for each phoneme in a pronunciations database consisting of 18500 American surnames. Using the phoneme occurrence statistics in conjunction with the posterior probability will provide more information towards selecting the correct phoneme sequence for a given proper noun. Since the occurrence of a particular phoneme sequence is a characteristic of the language of origin, inclusion of the phoneme sequence probability in the maximum likelihood framework allows application of linguistic knowledge sources.

It was described in Equation (19) that the probability of the phoneme sequence constituting the pronunciation of a proper noun can be modeled as a m^{th} order Markov model. The equation is reproduced here for convenience.

$$p(o_1, o_2, \dots, o_M) = \prod_{k=1}^M p(o_k / o_{k-1}, o_{k-2}, \dots, o_{k-m}) \quad (50)$$

This formulation is analogous to the concept of an n-gram language model [31] used extensively in speech recognition applications to represent patterns of word sequences occurring in the linguistic structure, where a history of the n preceding words is used to determine the likelihood of occurrence of the word currently under consideration.

The probability of a phoneme sequence is calculated in a manner similar to that of n-gram estimation. As before, all the distinct phonemes in the pronunciation database are indexed to integers in the range 1 through V , where V is the size of the phoneme set. Assuming a $m = 1$, i.e. a first order Markov process, the occurrence of a phoneme is only a function of the previous phoneme in the pronunciation. This will be denoted as a phoneme bigram, to maintain consistency with the speech recognition nomenclature. Then for each adjacent pair of phoneme indices x and y , the total number of occurrences of x following y (denoted by $n(x/y)$) is counted. Then the total number of occurrences of an individual phoneme x is given by

$$n(x) = \sum_{y=1}^V n(x/y) \quad (51)$$

and the total number of phoneme occurrences in the database is given by

$$n_{total} = \sum_{x=1}^V n(x) \quad (52)$$

Then, the bigram probability of a phoneme x given the preceding phoneme was y

is given by

$$p(x/y) = \begin{cases} \alpha \frac{n(x/y)}{n(x)} & \dots n(x/y) > 0 \\ \frac{1}{V} & \dots n(x/y) = 0 \\ f & \dots \text{otherwise} \end{cases} \quad (53)$$

Here, f is a lower bound on the probability and is set to some small value to avoid underflow. α is a normalizing constant used to ensure that all probabilities sum to 1 i.e.

$$\sum_{y=1}^V p(x/y) = 1 \quad (54)$$

If a phoneme sequence does not occur at all in the training data, then it is assigned the uniform distribution term $1/V$. If a large number of such cases occur in the bigram estimation, the performance of the system suffers since it approaches the uniform distribution system described earlier in this section.

As an alternative, a back-off bigram [32] can be constructed to overcome the sparse data problem to some extent. Here, if a particular phoneme sequence does not occur in the training data, its likelihood can be computed using a back-off weight of the history phoneme and the unigram probability (i.e. the probability of the individual phoneme) of the current phoneme. The unigram probabilities are defined as

$$p(x) = \begin{cases} \frac{n(x)}{n_{total}} & \dots n(x) > u \\ \frac{u}{n_{total}} & \dots \text{otherwise} \end{cases} \quad (55)$$

where u is the lower bound on the occurrence count of a single phoneme. The back-off bigram probability is given by

$$p(x/y) = \begin{cases} \frac{n(x/y) - D}{n(x)} & \dots n(x/y) > t \\ b(y)p(x) & \dots \text{otherwise} \end{cases} \quad (56)$$

Here, D is the discounting factor [33] used to deduct some of the available probability mass from the more frequent bigrams and distributing it among the low-occurrence bigrams, and is typically set to 0.5. t is a bigram count floor limit. The back-off probability is computed as

$$b(x) = \frac{1 - \sum_{y \in B} p(x/y)}{1 - \sum_{y \in B} p(y)} \quad (57)$$

where B is the set of all phonemes for which the bigram (x/y) exists. This ensures the validity of probability sum —

$$\sum_{y=1}^V p(x/y) = 1 \quad (58)$$

The equations for higher order phoneme n-grams can be derived in a similar fashion. However, an order larger than 2 or 3 is usually found to be impractical due to lack of sufficient training data, in which case most probabilities are assigned as back-off.

Since in the n-gram model of the phoneme sequence the likelihood of a particular phoneme depends on its position in the sequence, the information about whether a

phoneme can be at the start or end of a proper noun pronunciation is also important. However, the first phoneme has no preceding context, and the last phoneme cannot be identified *a priori*. Therefore, for the phoneme string representing the pronunciation of each proper noun in the training database, a start phoneme tag is assigned to the first phoneme and an end phoneme tag assigned to the last phoneme. This allows the likelihood of the phoneme being at the noun start or end is also captured in the n-gram model.

CHAPTER III

REVIEW OF CURRENT TEXT-TO-SPEECH TECHNOLOGY

Automatic generation of a list of probable pronunciations of proper nouns has application to both speech recognition and synthesis. In recognition systems, such lists can be used to build better pronunciation models for proper nouns, thus increasing the likelihood of obtaining accurate acoustic matches for such words. Similarly, a speech synthesis system can be tuned to produce a more natural utterance of the names required to be output.

To that effect, the aim of the proposed research is to develop maximum likelihood techniques that can automatically generate an accurate (i.e. acceptable with a high probability score) list of pronunciations for proper nouns.

The motivation for the proposed work originates from experiences at Texas Instruments in the late 1980s with attempts to field speech recognition technology in medical applications [34]. In many such applications, the ability to recognize a physician's or patient's name is crucial in providing a usable interface. For example, using numbers to access patient records is problematic due to the impracticality of remembering such numbers for any significant class of patients. Name recognition is a vital step in transforming medical record access from keyboard input to voice input. A comparable problem involving company names and product names exists in voice interfaces for advanced telecommunications services [35, 36].

Historical Overview

While the fields of speech recognition and synthesis have been active for several decades now, the focus on proper noun pronunciations is relatively recent. An extensive study was performed in the late 1980s to evaluate the accuracy of text-to-speech systems on pronouncing proper nouns [37] for a directory assistance application. In this work, it was shown that extensive handwritten rule sets were required to generate an accurate pronunciation of names. Such rule-based systems, even though deemed fairly accurate for the specific application, generated only the single most likely pronunciation for each proper noun.

Since many proper nouns have a number of highly probable pronunciations which can be rarely differentiated from the context of the application, a system generating only the single-most likely pronunciation essentially attempts to solve an ill-posed problem. Also, for a successful speech recognition application it is important that all plausible pronunciation alternatives be available to the system to build better quality acoustic models for the proper nouns.

A commercial product called DECTalk [38] was developed in the mid-1980s that converted unrestricted English text into speech, using a set of phonological rules and by handling exceptions with a lookup table. DECTalk used two methods for converting text into phonemes — first a word was looked up in a pronunciation dictionary of common words. If it was not found there then a set of phonological rules was applied to obtain a phonetic transcription along with stress assignments. These were then converted into speech sounds using transition rules and digital speech synthesis. For novel words for

which a correct pronunciation could not be obtained, the dictionary and rule sets required updating through explicit intervention. This approach was found to be highly labor-intensive and very limited in scope even for its intended application.

Many commercial systems, such as Bellcore's Orator [39], Bell Labs' TTS [40] and Mitsubishi's Anapron [41], follow variations on the same concepts of rule-based and/or dictionary-based look-ups to generate pronunciations for proper nouns. A major drawback of such systems is that they require an extensive set of handwritten letter-to-sound rules which make the systems cumbersome and expensive to develop and maintain. Moreover, such rule-based systems are constrained by their ability to generate only one pronunciation for a given proper noun, and do not generalize gracefully when presented with names not covered by the rule set.

Alternative approaches have emerged since then that employ statistical techniques such as Hidden Markov Models (HMMs) [42] and artificial neural networks (ANNs) [43] to model the stochastic distribution of pronunciations with respect to the letters in the spelling of the word. These have been met with varying degrees of success.

The general idea of applying neural network techniques to the text-to-speech problem is loosely based on a feasibility study performed in the mid-1980s that focused on automatically learning letter-to-sound mappings using neural networks [44]. As part of this study, a multilayered neural network model called NETtalk was developed as an alternative to DECTalk. It successfully demonstrated that a relatively small network can capture most of the significant regularities of the pronunciations of regular English words, as well as absorb a large number of the irregularities. It also had the advantage of being

language independent (i.e. it could be trained to be used on any language) and directly implementable in hardware. However, it was found to be limited in its ability to handle ambiguities that require syntactic and semantic levels of analysis.

In the late 1980s a technique for voice recognition of proper nouns using text-derived recognition models [45] was proposed and subsequently patented at Texas Instruments. In this technique, an algorithm was proposed to automatically derive recognition models from the text-only spelling of the name (rather than voice data containing nominal pronunciations). This system relied on a particular class of neural networks known as a Boltzmann machine [46], designed to generate multiple outputs for a given input. Such a network transforms the spelling of a proper noun to a network of distinctive features [47] describing articulatory movements required to produce various pronunciations of the name. However, this system was never implemented or evaluated, and has served as the starting point for the research proposed here.

Recently, statistical decision trees (DTs) have emerged as a viable technique for performing such nonlinear classification tasks with high degree of accuracy. For example, a technique that uses decision trees to automatically generate detailed phonetic pronunciation networks from a coarse phonemic transcription [48] has been developed at AT&T. Since each terminal node in the tree can store a statistical distribution of the phoneme associated with it, some decision tree based systems are also capable of generating more than one pronunciations.

Many of the systems described here have reported reasonable performance on standard English words. However, at present there exists no system that can automatically

and effectively model the peculiarities of proper noun phonology to generate multiple likely pronunciations for them.

The approaches to model pronunciations of words (including proper nouns) can be broadly classified into two categories — rule-based and data-driven. Rule-based systems employ a set of explicit (and often hand-crafted) rules designed by linguistic experts to determine the pronunciation of the input word. Data-driven methods, on the other hand, assimilate the statistical relationships between the features of the words (such as spelling, speech samples, formant tracks etc.) and the corresponding phonetic variation.

A discussion of various methodologies as applied to generation of pronunciations for proper nouns as well as regular words is presented here. A comparison of several state-of-the-art name pronunciation systems can be found in [49, 50].

Rule-Based Pronunciation Generation

The rules for generating pronunciation of a word are highly stylized and simplified approximations to the phenomena of natural speech, and are typically based on the morphological and phonological structure of the word [51]. In case of regular words in a language (such as English), the translation of the morphology of a word into a phonetic representation is fairly straightforward, and an appropriately large rule set is sufficient to provide coverage of a bulk of the vocabulary.

However, for proper nouns such as surnames the letter-to-sound relationship is exceedingly complex; and correspondingly the number of rules required to provide adequate coverage over a representative name set is impractically large. A typical TTS

system specialized for name synthesis may have an error rate of about 20% in terms of the coverage provided by its rule set when evaluated on a random set of names [52]. Most present-day systems address this problem using a two-fold strategy — very large rules sets are built to obtain coverage of approximately 60% of the surname distribution, and a specialized dictionary is used to look up the pronunciations for the remainder of the names.

The foundation of the rule-based systems lies in the observation that in most languages such as English, historically the alphabetical spelling representation of a word was closely related to its intended pronunciation [53, 54]. Over time, the pronunciations evolved to reflect sociolinguistic changes such as interaction with other languages [55] and resulted in such complex conventions as compound letters (for instance “ch” and “sh”), silent letters (e.g. “Psychology”) etc. [56].

Standard Rule-Based Systems

Early systems employing rule sets for pronouncing words exploited the fact that the pronunciation of a letter or a pair of adjacent letters in the spelling of a word is closely related to the adjacent letter context [57, 58]. Thus, rules were devised that would convert letter n-tuples such as “ph”, “ee” etc. to the corresponding phoneme. A subsequent set of conversion rules then assigned the remaining single letters to phones. The rules were usually ordered to treat consonants first and use the context specification provided by the consonants to convert vowels into phonemes.

From a perspective of text-to-speech synthesis, it was also found desirable to

generate the stress patterns for the pronunciations of these words. This was achieved by including stress rules such as the Chomsky-Halle set [59] used in [60], or the Hill and Nessly rule set [61] used in [62]. A majority of TTS systems that employ letter-to-phoneme rules also include stress pattern information for the syllables of that word. However, since the focus of this research is to generate pronunciation models that can be used both for recognition and synthesis, generation of stress patterns is not a critical issue. Therefore, a detailed discussion on application of stress rules is omitted here.

Due to the inherent variations in the letter-to-sound mapping of the language, even a large set of rules fails to accurately predict the pronunciations of the different words. By evaluating a large set of words on a purely rule-based system a list of exceptions can be generated, and the pronunciations of these words stored separately in the system.

An exception-handling pronunciation dictionary is necessitated by two factors for spelling-to-pronunciation conversion in typical TTS applications. First, a small number of words — approximately 2000 — is sufficient to cover almost 70% of words regularly used in English [63]. With a pronunciation dictionary for these words built into the system, a majority of the input text does not need to be treated with the letter-to-sound rules. Secondly, by adding other relatively frequent words that fail to be pronounced correctly using the rules to the dictionary, the performance of the system can be improved considerably [64, 65].

However, the size of the exceptions dictionary grows inversely proportional to the quality and extent of the letter-to-phoneme rules. Thus the storage requirements on such

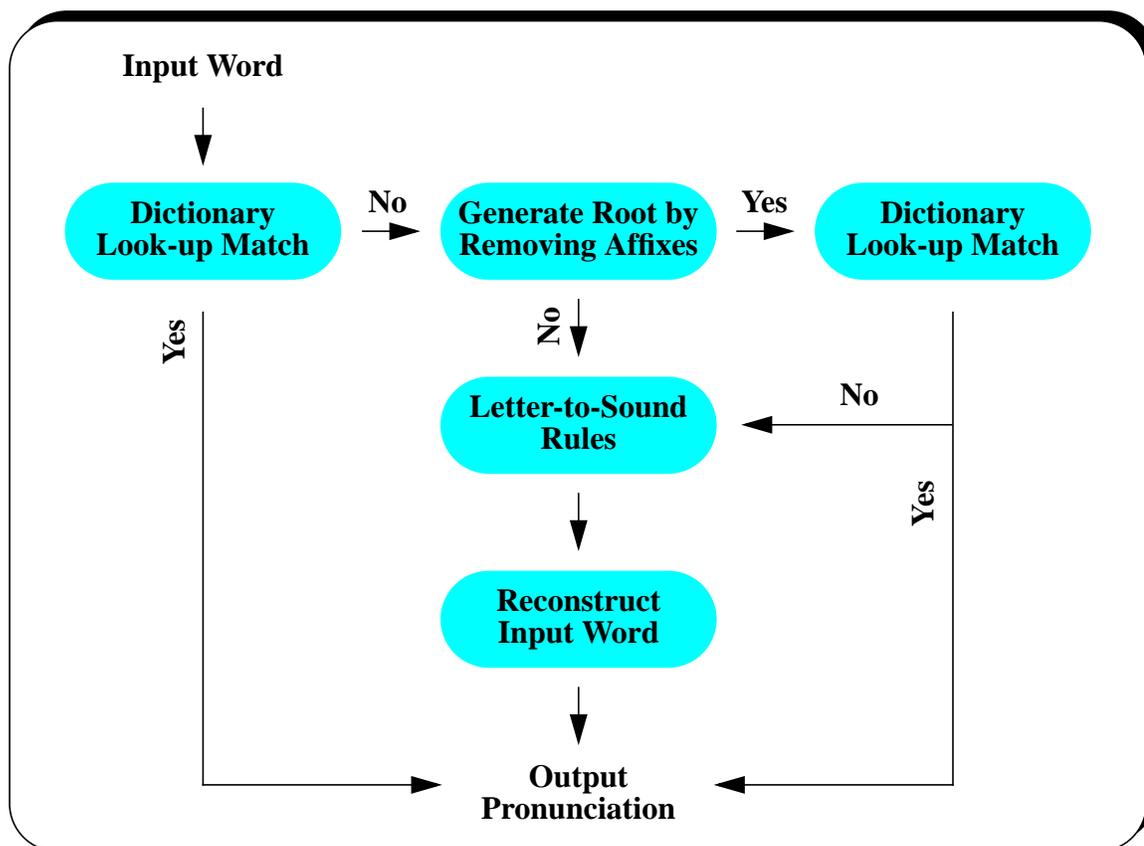


Figure 9. Schematic procedure for generating pronunciation of a word from its spelling using letter-to-phoneme rules and dictionary lookup.

rule-based systems are immense — either the rule set needs to be large to provide accurate coverage of different letter contexts, or the dictionary gets heavily populated to compensate for the deficiencies of the rules [66].

The standard procedure for generation of a pronunciation is depicted in Figure 9. First, the input word is looked up in a small pronunciation dictionary built into the system for handling exceptionally difficult words. If no match is found, the word is progressively broken into smaller parts called *morphemes* or *graphemes* by removing common prefixes and suffixes and recovering the root form of the word. If the root word does not match the

dictionary, the word is subject to the letter-to-phoneme rule set to generate its pronunciation.

Morpheme-Based Rules

An alternative to bulky exception-handling dictionaries is to maintain a pronunciations database of morphemes. Morphemes are often described as the smallest meaningful unit of language [67]. Morphological analysis of words is important when dealing with silent letters like “t” and disambiguating pronunciation of compounds such as “hh” in “hitchhike”, since a rule-based system will break down on such words. The benefit of morpheme-based methodologies is also supported by studies [68] which indicate that good spellers seek out morphemes in words and reconstruct the word spelling by concatenating its morphemic components.

The benefits of morpheme-based decomposition of words to generate pronunciations was demonstrated early on in [69] with a small dictionary of 3,000 morphemes. This was extended in [70] where a dictionary of 12,000 morpheme pronunciations was generated by interactive examination of the text of the Brown corpus [71], and augmented with a set of heuristic scoring procedures that will select the most plausible of the different possible morphemic parses of each word (e.g. “scar” + “city” vis-a-vis “scarce” + “ity” for the word “scarcity”).

The MITalk system [72] was built around such a morpheme decomposition algorithm with approximately 98% accuracy for words in regular text in addition to pronunciation rules. A similar system at Bell Labs [73] used a dictionary of 43,000

morphemes augmented with rules for stress assignment and affix analysis.

The most obvious advantage of morpheme-based pronunciation dictionaries is that a relatively small number of morphemes is required to provide coverage of a fairly large vocabulary, thus significantly reducing the memory requirements of the exceptions dictionary.

Proper Noun Systems

With sufficiently large rule sets and exception dictionaries, the rule-based systems described above performed fairly (accuracy of 95% at word level) well on generating pronunciations for general English text. However, their performance on predicting proper noun pronunciations degraded significantly — the early pronunciation systems geared specifically towards proper nouns [74, 75] had an error rate of more than 20% [76].

A study of the performance of contemporary TTS systems on pronouncing proper nouns [77] suggests that names often represent phenomena that pose problems for general letter-to-sound rules. Particularly, a large number of proper nouns have origins in languages other than English and therefore the letter-to-phoneme rules derived for English text do not translate well to their pronunciation.

One approach to overcome this problem is to first identify the language of origin for the name and then apply the rules specific to that language to generate the pronunciation. Language identification can be achieved by using the statistics of the frequency of occurrence of letter sequences in different languages and comparing these against the input name [78]. Therefore, adapting a system to proper nouns involves

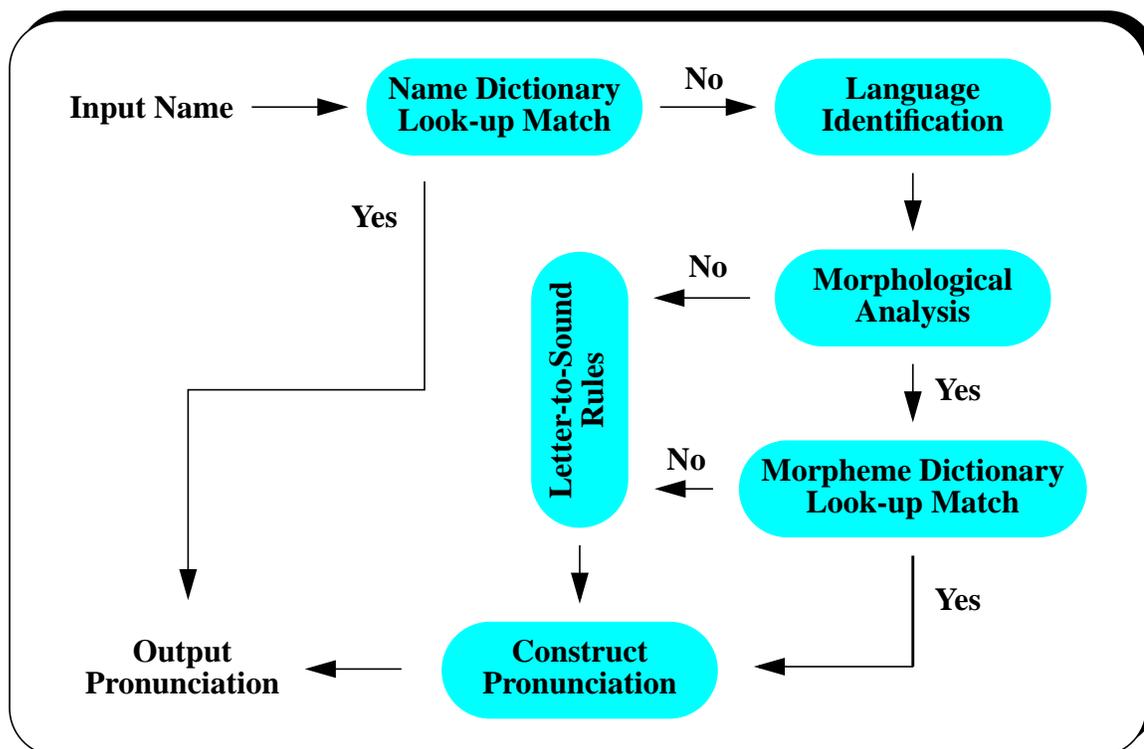


Figure 10. Schematic procedure for generating pronunciation of a proper noun from its spelling using dictionary lookup, morphology and letter-to-phoneme rules.

hand-crafting a very large number of restrictive letter-to-phoneme rules and a significant increase in the size of the exceptions dictionary. Most state-of-the-art commercial systems specially designed to generate proper noun pronunciations employ this strategy with some modifications. Figure 10 describes the general procedure for rule-based systems to predict pronunciations of proper nouns.

In the Orator system [39] from Bellcore, the name input to the system is first compared against a small exception dictionary of approximately 2500 names. If the name is not found in the dictionary, the system characterizes its language of origin and then subjects it to morphological analysis. A morpheme pronunciation dictionary is used to

determine the pronunciation of each morpheme. If the morpheme does not exist in the dictionary, it is subjected to letter-to-sound rules. The rules were specifically designed for pronunciation of names, taking into account factors such as the language, morpheme boundaries and orthographic context. The rule set adheres closer to the Americanized pronunciations of the names from foreign languages rather than the native pronunciations. Since the rule set is fairly large, for efficiency reasons it is compiled into a finite-state machine at run-time.

The DECvoice II system [79] from DEC is built around a modernized version of the DECTalk system [38] and follows a sequence of operations similar to the Orator to predict the pronunciation of a name. The first pass consists of a dictionary look-up, which is then followed by language identification if the name is not contained in the dictionary. A two-step process comprising of a set of filter rules followed by a trigram analysis of letters is used to characterize the language. The filter rules take into account characteristic letter patterns in the name to eliminate non-candidate languages. The trigram analysis is analogous to that described earlier in this section and in [78], and is required only if the filter rules yield multiple candidate languages. Pronunciation rules specific to the identified language are applied next to generate the pronunciation for the name.

The TTS system [40] from Bell labs relies heavily on a hierarchy of dictionary look-up methods. The first and simplest of these is to search for the name in a pronunciation dictionary of the 50,000 most common names found in the US. If this direct look-up fails, it applies morphological analysis-synthesis techniques to find names in the dictionary that are somehow related to the input name. These relationships include

rhyming analogies (e.g. “Mark” and “Stark”), appending or removing stress-neutral affixes to/from a name to match a dictionary entry (e.g. “Robinson” - “son” = “Robin”) or exchanging affixes (e.g. “Augustino” - “ino” + “elli” = “Augustelli”). The pronunciations for each of these morphemes are looked up in the dictionary and the final name pronunciation is constructed through similar operations. In the rare case where all such dictionary-based techniques do not provide a satisfactory match, a rule-based system is used to generate pronunciations for the exceptional morphemes.

The Anapron system [80] enhances the performance of a rule-based system using case-based reasoning by drawing analogies to existing cases [81] in the pronunciation dictionary when deriving letter-to-sound rules for an unknown name. It uses a dictionary of 5,000 surnames, which consist of the 2,500 most common American surnames and 1,250 surnames each sampled at random from the 2,500 through 10,000 and 10,000 through 60,000 most frequent surnames in the US. Conversion to phonemes is performed as a combination of rule-based and case-based reasoning. First, the system performs language identification and corresponding morphological analysis of the input name. For each morpheme, letter-to-phoneme rules are applied to generate its pronunciation. The rule selected for a particular letter is used to index into the dictionary to retrieve names that represent exceptions to the rule. If a strong analogy exists between any of the retrieved names and the input name, then the pronunciation of the exception is used for that morpheme. Analogies are determined by applying similarity metrics to the two names, and through empirical verification that evaluates the generalization behind the analogy on other similar names in the dictionary.

Pronunciation Generation by Learning

An alternative approach to predicting pronunciations by rule stems from psychological studies that suggest that people learn letter-to-sound conversions not by memorization of rules, but using analogies with similar patterns of letter sequences in words whose pronunciation is already familiar to them [82]. An early implementation of this strategy on regular words, which coupled the information contained in the frequency of occurrence of such analogous words [83], was found to match human performance in predicting word pronunciations over 90% of the test cases.

A system based on such an approach of learning by analogy often needs to store a pronunciation lexicon of words [84]. When a novel word is presented to the system, it searches through the lexicon for morphologically similar words and subsequently modify their phonetic transcription in accordance with the spelling of the input word. This is sometimes referred to as “explicit analogy” [85]. Systems based on this technique are only marginally better than rule-based systems, since they are still memory-intensive and slow (they often need to make multiple search passes through the lexicon to generate a class of words similar to the novel word).

Alternatively, systems can extract “implicit analogies” or generalized information from a training lexicon of pronunciations [86]. This has inspired research into treating the problem of letter-to-phoneme conversion as one of statistical pattern recognition. An additional benefit of this approach is that the system can learn both the language and pronunciation dependent aspects of the lexicon [87], and therefore can be applied to multiple languages such as French [88] and German [89].

A letter-pattern learning algorithm was developed in the early 1980s [90] that defined feature sets related to random sequences of letters using a forward-backward training algorithm analogous to one used in speech recognition systems. This algorithm was trained on a lexicon of 50,000 regular English words to determine the optimal feature sets for individual phonemes using a seven-letter context, and correspondingly a set of probabilities was generated for a search tree recognition model. This system yielded a letter-to-phoneme conversion accuracy of 94% for a 5,000 word test set.

A statistically learning neural network was implemented in the NETtalk system [44] which accepted as input a similar window of seven-letter context, and output the phoneme corresponding to the middle input letter. It used a set of 120 hidden neurons, 29 letter symbols for the input and a set of 40 phoneme symbols to represent the output pronunciations. The network weights were initialized randomly and trained incrementally using simulated annealing [28] over a lexicon of 20,000 words. In a closed-loop evaluation (i.e. testing on the same set of words) this network was found to achieve 90% phoneme classification accuracy — comparable to a contemporary rule-based system without an exceptions dictionary. A similar network architecture was subsequently applied to Spanish, with comparable performance [91].

An algorithm was defined in [92] in which a decision tree was developed to perform comparison of substrings from the spelling of a word in an optimal and efficient fashion. A moderate-sized decision tree trained on a pronunciation lexicon of 20,000 words was constructed with half the data held out for evaluation. The performance for accurate classification of individual letters was found to be 93% on this held-out subset of

data. A similar system [93] automatically maps groups of letters to phonemes by learning from a training lexicon.

A review of the error modalities in these early classification systems reveals that such systems failed to generalize in an optimal fashion over different classes of letters and phones [51]. For instance, a vast majority of errors (about 80%) were found to occur during classification of vowels. This can be attributed to the extent of letter context that influences stress assignments and therefore the pronunciations of the vowels. In addition, since pattern matching systems require an output for every input, compound letters such as “sh” and compound words with letters that have no corresponding phones (e.g. “e” in “lifeboat”) posed problems of alignment.

Recently, new data-driven techniques such as stochastic phonographic transduction (SPT) [94] have emerged that model the spellings and pronunciations of English words as the output of a stochastic grammar which is derived from a pronunciation dictionary. The terminal symbols of this grammar are letter-to-phoneme correspondences, and the rewrite (production) rules of the grammar specify how these are combined to form acceptable spellings for English words and their pronunciations [95]. For a given word, its pronunciation is produced by parsing its spelling according to the letter-part of the terminal symbols, and selecting the best sequence of corresponding phoneme-part terminals according to some optimization criteria [96].

For practical systems that are efficiently trainable, the grammar is assumed to be regular and that the pronunciation of a word is modeled as a Markov process of concatenated letter-phoneme pairs. The SPT training algorithm then amounts to the

inference of an optimal set of correspondences and the estimation of their associated transition probabilities. Transduction to produce a pronunciation for a word given its spelling is achieved by Viterbi decoding [97]. A phoneme classification rate of 93% was reported using this technique on an open-loop test over regular English words.

Name Pronunciation Systems

Many of the statistical learning techniques described above have been attuned to generate pronunciations of proper nouns. For instance, a modified version of the NETtalk system classifies a phoneme for each letter of the input name, and then compares letter sequences of different lengths from the name to entries in a pronunciation dictionary [98, 99]. If a match is found, the dictionary pronunciation closest to the generated phoneme is used as the output.

Similarly, the TTS systems from AT&T [100] employ stochastic decision trees to derive name pronunciations from coarse phonemic representations generated from a base letter-to-sound pronunciation rule set.

The text-to-phone converters developed at CMU [101] use pronunciation dictionaries to train a system that produces a set of alignments between the letters and phonemes given a set of word and its pronunciation. These are stored as an ordered list of symbols and converted into feature vectors, one per letter, that include a given amount of context. A stochastic decision tree is constructed based on the feature that splits the data into the purest pair of subsets. Here, each feature vector can be thought of as a string generated by a context-sensitive regular grammar. Therefore, the decision tree acts as a

finite state transducer that maps the letter context to corresponding phonemes [102].

Modeling of Phoneme Sequence Probabilities

As described earlier (e.g. Equation (21)), the application of phoneme sequence probabilities has a significant influence on the maximum likelihood prediction of the pronunciation of a proper noun. The parallelism between the language modeling problem in speech recognition and the phoneme sequence modeling for proper noun pronunciations is obvious. Both problems deal with estimating likelihoods of units based on their position in a sequence — words in the former case and phonemes in the latter. Therefore, the techniques applied for language model estimation can also be used to model the phoneme sequences.

The choice and scope of the phoneme sequence model has a significant influence on the performance of the pronunciation generation system. It provides constraints on the occurrence of particular phonemes and phoneme patterns depending on the nature of the proper noun and its language of origin. Thus it plays a considerable role in determining the search space for generating the best pronunciation for the given spelling.

Since the order of occurrence of phonemes in a pronunciation does not follow a formal grammatical structure, the phoneme sequence probabilities need to be estimated statistically over a large number of proper noun pronunciations.

Static Models

A phoneme sequence model can be used by a pronunciation generating system to

apply various levels of linguistic constraints. A *uniform* model that assigns equal probabilities to all phonemes in the dictionary does not impose any constraint on the system and therefore is not very useful. As described in Chapter II, n-gram models [103] provide information about the identity of a phoneme given its history i.e. the sequence of the preceding n phonemes in the pronunciation.

$$p(o_1, o_2, \dots, o_N) = \prod_{k=1}^N p(o_k / o_{k-1}, o_{k-2}, \dots, o_{k-n}) \quad (59)$$

A value of n greater than 3 is not practicable for implementation, and hence n is typically limited to 2 (bigram model) or 3 (trigram model). A trigram is better than a bigram model as it carries more information, but requires more training data.

The n-gram model is simple yet powerful [104], but it is static. It uses only the very immediate history of the phoneme and does not depend on or vary with the data being observed. Therefore it is not capable of adapting to a change in the linguistic style and cannot exploit this to enhance the probabilities of related phonemes while suppressing those of others.

Dynamic Models

An adaptive or dynamic model improves upon the performance of a static n-gram model by changing estimates of the phoneme probabilities depending upon the part of the pronunciation observed so far. This is particularly useful if a model trained on data pertaining to a specific domain is used in another domain, as the model can adjust to the new language.

Also, if the pronunciation dictionary used for training can be partitioned into different root languages, then an adaptive model trained on this heterogeneous source can exploit the sub-language structure to improve performance. There are a number of techniques used in language modeling that implement this approach to capture these linguistic phenomena, and can be applied similarly to phoneme sequences.

- *Long-distance n-grams*: These are similar to conventional n-grams except that they precede the phoneme o_k by q positions [105].
- *Triggers*: A *trigger pair* [106, 107] consists of two phoneme sequences where the occurrence of one changes the probability estimate of the other. Constructing a trigger model involves eliminating all pairs of phoneme sequences that are not significantly related. The effects of several triggers towards the triggered sequence are combined and the trigger information is integrated with the static model in a way that preserves the advantages of both. A maximum entropy (ME) algorithm [108, 109] is used to train the trigger-based model. While the ME approach is intuitively simple, easy to implement for a variety of problems and guaranteed to converge to a solution, it suffers from very high requirements of memory and computation and does not have a well-defined rate of convergence.
- *Cache Models*: Once a phoneme (or phoneme sequence) occurs in a document, its likelihood of recurrence is greatly increased. This tendency is most true of rare phonemes, and reduces as the phoneme becomes more frequent in occurrence. Based on this phenomenon, the last L phonemes (or phoneme

sequences) of the pronunciations seen so far are stored in a cache. This cache is used to estimate the dynamic unigram, bigram and trigram probabilities [110, 111, 112] and then incorporated with the static model using interpolation techniques. Caches can also be formed based on the number of times the phoneme o_k already appeared in the history and based on distance i.e. the last time o_k occurred in the history.

- *Class Grammars*: Instead of phonemes, classes of phonemes are taken as the units of the model. The probability of phoneme occurrence is determined by the probability of occurrence of that phoneme class [113].
- *Tree-based models*: These models [114] generate a binary decision tree from the training data to cluster phoneme histories. Each node of the tree is associated with a state of the model and each leaf corresponds to a valid phoneme sequence. The tree is constructed using yes/no questions that reduce the uncertainty of predicting the next phoneme at every node and thus minimize the average entropy at every leaf. However, these methods are computationally expensive.
- *Mixture models*: The phoneme sequence model is built as a mixture of several component models, each of which is trained on the n-gram statistics of a particular class of names. The component models can be combined using either dynamic-weight mixtures at n-gram level [115] or static-weight mixtures at the complete pronunciation level [116, 117]. The proper noun classes can be

specified by hand, or can be determined automatically using clustering techniques. Robustness of parameter estimation for mixture components is an important issue here as each component model is trained only on a part of the available data.

Other Techniques

There are various other techniques for language modeling with different properties that can be used to model phoneme sequences. For instance, while context-free [118] and unification [119] models do a more realistic job of capturing the distribution of the phonemes, they are cumbersome and computationally expensive. On the other hand, finite state [120, 121] models that try to model all pronunciations in a single network provide more constraints on the distribution, but do not provide as realistic a representation of the phoneme statistics.

Observations

The performance of the learning systems in accurately classifying letters to phonemes has been comparable, if not worse, compared to the rule-based systems on regular English text-to-speech applications. In case of proper nouns, the performance gap is much wider. This is primarily due to the difficulty in capturing the stochastic relationships between letters and phones which are inherently more complex for proper nouns, since the systems require an appropriately representative training data set to optimally generalize the letter-to-sound inferences.

In practice, most present-day commercial speech synthesizers rely on

pronunciation rules, morphological analysis and exception look-up to create a letter-phoneme alignment for the given names. Statistical techniques are then used to model allophonic variations [122, 123], and to expand the preliminary alignments into more refined pronunciations [124, 125] for better synthesis of the phonemes into speech output, and to construct better pronunciation models for automatic recognition.

However, the data-driven properties of statistical systems make them amenable to automatic updates with new training, and their strong theoretical foundations that have been successfully applied to most other aspects of computerized processing of speech warrant further research into their merits for generating pronunciations of proper nouns. Moreover, all the rule-based systems described in this chapter deal with a single pronunciation for each input word. Since the stochastic learning systems have the ability to generate probabilistic estimates of the plausibility of different pronunciations for a given letter context [126], research in developing algorithms to automatically generate likelihood-ordered multiple pronunciation lists for proper nouns is particularly attractive.

CHAPTER IV

PRONUNCIATIONS DATABASE

For a speech recognition / synthesis system to learn to generate a list of proper noun pronunciations, the most obvious way would be to train the system on audio data that exemplifies the different pronunciations for each name. However, no speech database that provides comprehensive coverage of a range of proper nouns exists, and it is extremely expensive to construct.

A resource that is readily and inexpensively available for most proper nouns (especially personal names, surnames, places etc.) is an orthographic transcription or spelling. It follows that an ability to train a voice interface system to automatically generate multiple pronunciations for a given spelling of the name will be greatly beneficial. Such a training methodology needs to exploit the statistics of the input data to capture the inter-relationships between letters and sounds. Therefore, a database that consists of a large number of proper nouns and their possible pronunciations is needed to train the system.

Numerous data has been collected anecdotally on the problem of alternate pronunciations of proper nouns, and there exist some proprietary databases that cover restricted domains of proper nouns [128, 129]. However, none of this data has ever been incorporated into a publicly available database. Thus there does not exist an appropriate development database that provides comprehensive coverage of various kinds of proper nouns and their pronunciations.

As a result, a part of the effort expended in this research has been utilized for the development of a proper noun pronunciations database that has been placed in the public domain [130] for free distribution. Since a significant number of applications involving voice-driven interfaces (e.g. medical data entry, directory assistance etc.) involve names of people, the current phase of the database effort has been focused on surnames. The database currently consists of 18494 surnames (last names) found in the United States of America. A total of 25648 pronunciations have been transcribed for these names.

The surnames in the database represent a diverse set of proper nouns with numerous ethnic origins, source languages and dialectic variations. Therefore, the phone set spanning the English language is insufficient to provide full coverage of the variation in pronunciations. Moreover, a phonetic convention that uses ASCII symbols for all the phonemes facilitates transcription and subsequent processing of the pronunciations. From this perspective, the Worldbet phonetic convention [8] is an ideal choice for transcription of the pronunciations. It contains symbols for phonemes that appear in a large number of languages besides English, and these can be represented in a manner amenable to computerized processing. The subset of the Worldbet phoneme set used in the database, along with illustrative pronunciations is listed in Table 1.

Collection And Transcription

Construction of a representative database of surnames that can be compiled with reasonable requirements on resources of time and expenditure presents some peculiar problems. While the interesting pronunciation problems occur in the outliers of the

surname distribution — names occupying the bottom 0.01% of the total, the distribution is also extremely skewed. The 2000 most common surnames account for about 15% of the population, while the remaining 85% of the population covers almost 1.5 million

Phoneme	Example	Phoneme	Example
i:	heep	I	mill
E	bell	@	adams
A	aanstoos	^	trump
>	allman	&	alba
U	pool	u	bookman
aI	pine	ei	mate
>i	joy	aU	cloud
oU	close	iU	pure
&r	easter	p	peabody
b	baggs	t	stokes
d	hardy	k	culkin
g	gaston	h	hermit
v	vail	D	worthy
T	roth	s	sandler
z	rose	S	nash
Z	leisure	f	fielding
m	ames	n	inez
N	golding	dZ	johnson
tS	chow	l	calahan
9r	robinson	j	young
w	willis	ks	fox
&k	mcDowell	–	bateman

Table 1. A list of phoneme symbols used for representing the pronunciations of proper nouns.

surnames. In our database we have strived to achieve a reasonable mix of common names, names with infrequent occurrence and names that are known to present problems for letter-to-sound conversion because of complex morphology or difficult stress assignments.

A significant fraction of the surnames included in this database was obtained from a study performed at Texas Instruments [37], which developed a small set of surname-pronunciation pairs to evaluate the intelligibility of text-to-speech synthesis systems on surnames. A few of the sources which contributed significantly to the list of surnames covered by this database are enumerated below:

- The 2,000 *most common surnames* from the Social Security Administration records compiled in 1964 [131].
- A list of *medium frequency surnames* obtained from a random sampling of the 2,000 to 50,000 most common names in the 1964 Social Security Administration records. This list was provided by M.F. Spiegel, Speech and Image Processing Research Division, Bell Communications Research.
- Some *challenging surnames* that look for somewhat subtle distinctions among letter sequences whose pronunciation differs significantly based on the remainder of the name (e.g. “bos” in “Bostwick” and “Bose”); and/or common patterns that appear in some foreign names (e.g. “Stein”, “Renoir” etc.). A list of such names was provided by M.F. Spiegel, Speech and Image Processing Research Division, Bell Communications Research.
- Surnames that are *morphologically complex* and require knowledge and/or analysis of the morphology of the name for an appropriate pronunciation (e.g.

“Holinshead” can be mispronounced as “Holin Shead” if the “head” morpheme is not identified correctly and separated for pronunciation). A sampling of these was also provided by M.F. Spiegel, Speech and Image Processing Research Division, Bell Communications Research.

- A list of *common but interesting names* that demonstrate the prevalent problems in generating automatic and accurate pronunciations was also supplied by M.F. Spiegel, Speech and Image Processing Research Division, Bell Communications Research. This includes foreign names (e.g. “Subramanian”), morphologically complex names (such as “Blankenship”) and names with unusual stress assignment (e.g. “Cadwallader”).
- A number of last names were generated at Texas Instruments as those belonging to the people working at Texas Instruments Computer Science Center circa 1990.
- A list of surnames was extracted from electronic phone books compiled by the Naval Research Laboratory. Another was created at MIT from people’s user names on the MIT computers.
- A list of about 2000 surnames not already included in the previous lists was obtained from the “30000 names” database being developed at the Oregon Graduate Institute [132].

Automated transcription of these surnames would have required a commercial rule-based text-to-speech synthesis system. A disadvantage of this approach, besides the cost involved, was that this would have yielded only a single pronunciation for each name.

Moreover, accuracy of these pronunciations was an important issue; especially for the more complex and unusual names. Converting the generated pronunciations to the Worldbet phoneme set would introduce additional complications in the process.

Therefore, the phonetic transcription was performed by hand where each name was transcribed multiple times to obtain all the correct pronunciations possible. No effort was made during this process to align letters or letter n-tuples in the spelling of the surname to the corresponding phonemes in the pronunciations.

Strict quality control was maintained throughout the transcription process. Linguistic experts were consulted for transcription of surnames with unusual characteristics, while automated scripts were employed to ascertain the veracity of the phone set used and to correct typographic errors.

Phoneme Alignment With Spelling

As described in Chapter II, the phonetic classification paradigm is designed to look at each letter of the input surname spelling one by one in context of its nearest neighbors, and output a phoneme symbol in accordance. Thus for training purposes it is necessary that there be a phoneme corresponding to every letter in the input spelling [133]. Since in many cases a single phoneme encompasses a group of letters or vice-versa, such one-to-one alignment is not possible for the pronunciations in their original transcribed form.

Situations where a single letter corresponds to more than one phoneme (e.g. “Max” is transcribed as “*m @ k s*”) appear relatively infrequently in the database, and

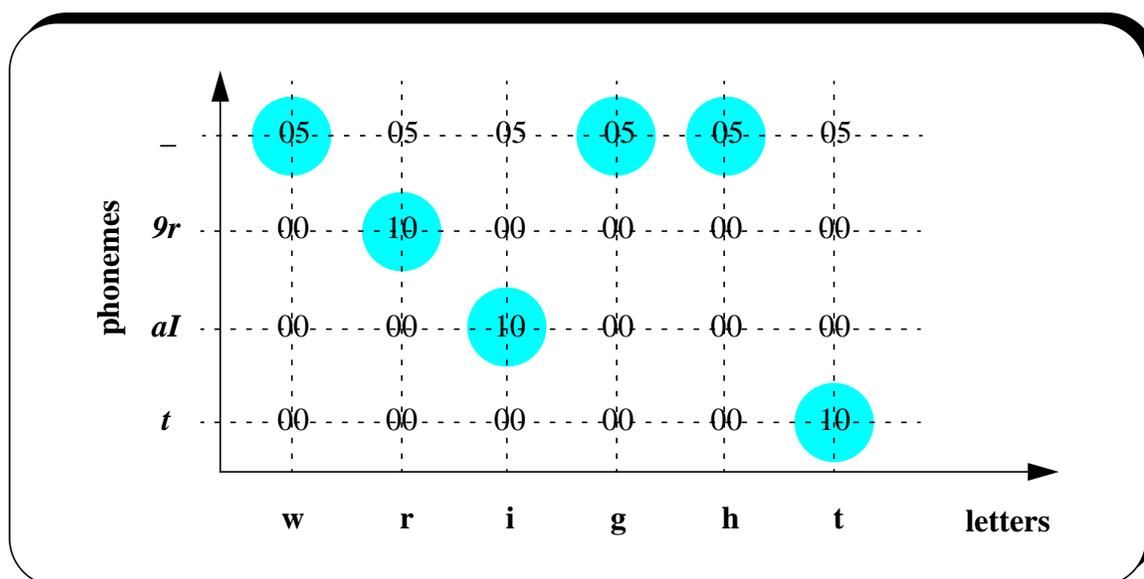


Figure 1. The dynamic alignment procedure illustrated for the word “Wright” and nominal pronunciation “*9r aI t*”. The aligned pronunciation is “*_ 9r aI _ _ t*”.

have been handled by using compound phones (such as “*ks*” for “*k s*”, so that “Max” is pronounced “*m @ ks*”).

On the other hand, the cases in which a grapheme (i.e. a sub-word unit or a group of consecutive letters) of the surname maps to a single phoneme (for instance, “Wright” is pronounced “*9r aI t*” where the two letters “wr” account for a single phoneme “*9r*”; similarly the letters “igh” together correspond to the one phoneme “*aI*”) are quite common. To correctly align such groups of letters in the spelling with the corresponding phonetic expression, a new *silent phoneme* symbol denoted as “*_*” has been introduced (therefore, “Wright” now gets transcribed as “*_ 9r aI _ _ t*”).

A dynamic programming algorithm has been developed to perform automatic alignment of the spelling-pronunciation pairs by introducing such a silent phoneme at appropriate positions in the pronunciation phoneme sequence. This algorithm assigns

numeric scores to all possible letter-to-phone maps. A phoneme corresponding to a group of letters is aligned with one of the letters according to a strategy that maximizes the total alignment score for the entire word. The other letters are aligned with the blank phoneme. An example of the dynamic alignment can be seen in Figure 1.

CHAPTER V

PROPOSED EXPERIMENTS AND EVALUATION

As described in Chapter II, the experimental framework for evaluating the maximum likelihood estimation techniques for automatic generation of pronunciation lists for proper nouns consists of a sliding window input of the context-dependent n-tuples of letters derived from the spelling of the name, and its classification into the most likely phoneme symbols. The output phonemes corresponding to each letter in the spelling will be then used to construct a pronunciation network for the name which represents the possible pronunciations along with the corresponding likelihood scores.

The surname pronunciations database described in Chapter IV will be used as the test bed for all experiments and evaluations. This database currently consists of 18494 surnames, which will be divided into two sets as follows:

- A *training set* consisting of 15000 names and the corresponding pronunciations will be picked randomly. The surname-pronunciation pairs in this set will be used to train the statistical models that capture the letter-to-sound dynamics of name pronunciations. Closed-loop evaluations (i.e. testing on this data) will be performed to demonstrate the ability of the system to learn from these exemplars.
- An *evaluation set* of the remaining 3494 surnames and their pronunciations will be held out for open-loop testing (pronunciation generation on previously unseen names). This will evaluate the ability of the system to generalize its

knowledge and extend it to unfamiliar inputs.

To cross-validate the results i.e. to ensure that the results are robust to the spurious artefacts in the training data (such as the order of the names in which the system is trained etc.), the above partitioning will be carried out three times. These partitions will be designed to create three mutually exclusive evaluation sets, and correspondingly the respective training sets will have about 60% overlap.

Error Metrics

Traditionally, a good measure of the performance of text-to-speech conversion systems has been the voice quality of the output and the intelligibility of the generated pronunciations [134]. Other factors that influence evaluation of a TTS system include listener fatigue, cognitive load [135] and response latency [7]. On the other hand, the effect of pronunciation modeling quality in a speech recognition application is most apparent from the quality of the acoustic matching and the recognition performance measured as the word error rate.

The performance of the pronunciation generation system will be measured in terms of the proportion of surnames for which the system is able to generate at least one accurate pronunciation. As the problem of text-to-phoneme conversion is essentially unrelated to the transformation of phonemes to voice output, synthesis-based criteria will not be employed to evaluate the quality of pronunciations.

For phonemic transcriptions of proper nouns to be useful for synthesis and/or recognition, it is important to represent the possible variation in the pronunciations of each

name. Therefore an error metric based on how well the generated pronunciations of a name span its reference pronunciations in the dictionary is proposed. Since many surnames have multiple reference pronunciations, and the system also generates multiple pronunciations for each surname, the performance measurement involves a many-many correspondence between the reference and hypothesis pronunciations. As a result, this error metric will consist of three measurements:

- *all correct* — all the pronunciations in the reference dictionary are generated by the system.
- *some correct* — at least one, but not all of the reference pronunciations are included in the generated pronunciations.
- *no correct* — none of the reference pronunciations are found to match any of the pronunciations generated by the system.

The *no correct* value represents the word-level error rate of the system in generating pronunciations. The higher the number in *all correct*, the better is the performance of the system. The sum of *all correct* and *some correct* names for the output of the system is an indication of its ability to generalize the letter-to-phoneme correspondence learnt during training.

Scoring Methodology

The performance of the system in predicting multiple pronunciations for the input names will be measured automatically using a scoring software that locates the reference pronunciations in the dictionary and compares them with the pronunciations generated by

the system. The resulting score file will thus contain each test case proper noun, the number of reference and hypothesis (generated) pronunciations for it and the number of correct pronunciations generated. The correctly hypothesized pronunciations and the incorrect pronunciations can also be listed in decreasing order of likelihood scores. At the end, the overall scoring statistics — the number of test case proper nouns, the number of total reference pronunciations, the number of pronunciations generated and the error statistics as described in the previous section will be calculated.

Initial Results

A set of preliminary experiments have already been conducted to demonstrate the feasibility of some of the techniques proposed in this research, such as artificial neural networks, statistical decision trees and a simple table look-up scheme based on the context interpolation method. These experiments follow the framework defined in Equation (16), and therefore do not assume any dependence of the predicted phoneme on the previously identified phoneme (i.e. a uniform distribution is assumed for the occurrence of a phoneme in the phoneme sequence).

Since the likelihood of the optimal phoneme sequence can be found by simply multiplying the probabilities of the individually optimal phonemes, no search is required to determine the best pronunciation for the given proper noun. Since the problem of modeling the letter-to-phoneme mapping is treated as a pattern classification paradigm, the phoneme classification error rate is the most significant factor contributing to the performance metric for this system.

Phoneme Classification Using Neural Networks

Pronunciation generation systems using Boltzmann machines — feed-forward backpropagation networks with a stochastic component — were first trained on a small subset of 1200 surnames using the standard simulated annealing techniques [28] and tested on a held-out subset of 400 names [136] in the surname pronunciation dictionary. A single and fixed context length was used, as opposed to a series of contexts. The *no correct* pronunciation generation error rate was found to be approximately 50% on an open loop evaluation, while the phoneme classification error rate was around 20%. When trained on the full training set, an open loop evaluation on the full test set yielded a *no correct* error rate of 65% and a phoneme classification error rate of 30% [137] using 1-best pronunciations i.e. a single output phone per letter.

Experiments conducted with neural networks having more than one hidden layer yielded no significant improvement over those with a single hidden layer, but caused considerable overhead in terms of the memory and time required for training.

While the neural network system was found to be highly accurate on smaller phoneme classification tasks, its performance degraded on larger-scale applications [21]. An analysis of the error modalities revealed some inconsistencies with the training database which have been rectified since then. With the improved training algorithms presented in Chapter II and a cleaner database, the performance is expected to improve.

Phoneme Classification Using Decision Trees

A parallel study that explored simple stochastic decision trees to classify letter

system	description	% error
Orator	dictionary lookup, language identification, rules	93.00
DECvoice	dictionary lookup, language identification, rules	93.00
TTS	progressively coarse dictionary lookup	89.00
Anapron	rules and case-based transcription	86.00
NETtalk	NETtalk with block-decoding post-processor	78.00
Neural net	multilayered feed-forward network	74.00
Decision tree	Bayesian criteria for node-splitting	82.00

Table 2. Comparative performance of various name-pronunciation systems on a 400 surname test set (courtesy [49]) compared to performance of test systems on a similar test set of 400 names.

strings derived from surname spellings into the corresponding phonemes was conducted [138, 139]. Using Bayesian estimates of node likelihoods in the tree, a system trained on 5-letter contexts of the names from the surname pronunciations database reported a *no correct* name pronunciation error rate of 38% on the evaluation set, and approximately 12% on a closed-loop evaluation [140].

Binary Tree Search / Table Look-up of Likelihoods

A system that generated all possible letter context sequences (up to a fixed length of context) present in the training data, and for each partial letter sequence stored counts of all the possible phoneme symbols in a binary search table [141] was developed and trained on the surname dictionary. During evaluation, for each letter in the spelling of the input name the longest matching partial letter sequence was found in the table, and the corresponding phoneme symbol with the maximum count was looked up and output. This

system, which uses the most elementary form of likelihood maximization, recorded close to 40% of *no correct* error rate.

A comparative performance evaluation of surname pronunciation generation using various text-to-speech systems is provided in [49]. Since the training and test data used for this benchmark is not publicly available, it is difficult to compare the performance of the algorithms discussed here against this benchmark. However, by selecting test data from the surname pronunciation dictionary in an analogous manner, the test conditions were approximated and a similar comparative evaluation was conducted. The results are presented in Table 2. Even though a valid comparison is not possible, the system performance was estimated to be comparable to many of the rule-based systems.

Proposed Evaluations

The proposed system designed to generate multiple pronunciations for proper noun spellings will be trained using the surname pronunciations training database to model the letter-to-phoneme *a posteriori* likelihoods using the various algorithms developed in Chapter II. The database will also be used to estimate n-gram models for the phoneme sequence likelihoods.

For evaluation, a dynamic programming search technique [142, 143] will be used to find phoneme sequences with the maximum cumulative scores obtained by combining the computed score of the input letter n-tuples against the letter-to-phoneme models, and the phoneme sequence probabilities. The letter-to-phoneme models explored will be variations of the stochastic neural network, as well as the context interpolation models. A

uniform phoneme distribution (i.e. no phoneme likelihoods included in the maximum likelihood estimation), as well as unigram and bigram models will be used for the phoneme sequence likelihood.

Evaluation of the Neural Network Model

The neural network system will be trained with letter n-tuples of multiple context lengths simultaneously. These strings will capture the context information from non to very near contexts (only immediately adjacent letters) through more distant features (2 or 3 neighbors on each side). The classification will be performed in the following different fashions.

1. Each input vector will be classified to a single phoneme along with the associated likelihood estimate, and a stochastic component with multiple applications of the same input vector will be used to generate the phoneme variants with different likelihoods.
2. For each input vector, the possible phoneme outputs will be simultaneously classified using a multi-output network. A threshold value on the likelihood estimates will be used to prune away poor-scoring alternatives and an N-best list of phonemes will be output.
3. As a variation on 2 above, the outputs of the network will be weighed with the *a priori* likelihood of the corresponding phoneme being mapped to the input letter sequence. The *a priori* distribution will be generated from the training data and this weighing will be used in both training and evaluation, or only

during evaluation.

4. A time-delayed version of the network, that receives a feedback input of the phoneme predicted for the previous letter string will be implemented to include the phonemic context in addition to the orthographic information.

Evaluation Using the Context Interpolation Models

Different sets of context interpolation models will be trained on the surnames database using a range of contexts lengths, and the corresponding back-off weights also estimated in the process. This will be done by generating all the partial letter sequences up to the full context length with the same central letter, and keeping counts of the frequency of occurrence of each phoneme. During the Viterbi search to determine the best phoneme sequence, these models will be used to estimate the likelihood of the input letter n-tuple against the models for each phoneme. Then a list of the highest-scoring hypotheses will be generated as the optimal pronunciations.

Finally, an evaluation of each system output will be performed using the scoring paradigms described earlier, and the performance of each system analyzed in comparison with the others.

CHAPTER VI

REFERENCES

- [1] M. Higgins, M. Higgins and Y. Shima, "Basic Training in Pronunciation and Phonics: A Sound Approach," *The Language Teacher*, Vol. 19, No. 4, pp. 4-8 and 16, April 1995.
- [2] K. Belhoula, "Rule-based Grapheme-to-Phoneme Conversion of Names," *Proceedings of the 3rd European Conference on Speech Communication and Technology*, Vol. 2, pp. 881-884, Berlin, Germany, 1993.
- [3] M.F. Spiegel, "Pronouncing Surnames Automatically," *Proceedings of the Voice I/O Systems Conference*, pp. 107-132, September 1985.
- [4] L. Lamel and G. Adda, "On Designing Pronunciation Lexicons for Large Vocabulary, Continuous Speech Recognition," *Proceedings of the 4th International Conference on Spoken Language Processing*, pp. 6-9, Philadelphia, PA, October 1996.
- [5] M.Y. Liberman, "Phonemic Transcription, Stress and Segment Durations for Spelled Proper Nouns," *Journal of the Acoustical Society of America*, Supplement 1, Vol. 64, S163, 1979.
- [6] S. Fitt, "The Pronunciation of Unfamiliar Native and Non-native Town Names," *Proceedings of the 4th European Conference on Speech Communication and Technology*, Vol. 3, pp. 2227-2230, Madrid, Spain, 1995.
- [7] D.B. Pisoni, "Effects of Talker Variability on Speech Perception: Implications for Current Research and Theory," *Proceedings of the International Conference on Spoken Language Processing*, pp. 1399-1407, November 1990.
- [8] J.L. Hieronymus, "ASCII Phonetic Symbols for the World's Languages: Worldbet," *Technical Report*, AT&T Bell Laboratories, Murray Hill, NJ, 1994.
- [9] L.R. Rabiner and R.W. Schaffer, *Digital Processing of Speech Signals*, Prentice Hall, Englewood Cliffs, NJ, 1978.
- [10] A.J. Viterbi, "Error Bounds for Convolutional Codes and an Asymptotically Optimal Decoding Algorithm", *IEEE Transactions on Information Theory*, Vol.

- IT-13, pp. 260-269, April 1967.
- [11] Y.L. Chow and R.M. Schwartz, "The N-Best Algorithm: An Efficient Procedure for Finding Top N Sentence Hypotheses", *Proceedings of the DARPA Speech and Natural Language Workshop*, pp. 199-202, October 1989.
 - [12] L. Nguyen, R. Schwartz, Y. Zhao and G. Zavaliagos, "Is N-Best Dead?," *Proceedings of the DARPA Human Language Technology Workshop*, pp. 386-388, March 1994.
 - [13] L. Breiman, J.H. Friedman, R.A. Olshen, C. Stones, *Classification and Regression Trees*, Wadsworth, Monterey, CA, 1984.
 - [14] S.R. Safavian and D. Landgrebe, "A Survey of Decision Tree Classifier Methodology", *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. 21, No. 3, pp. 660-674, May 1991.
 - [15] N. Deshmukh, A. Le and J. Picone, "Advances in Automatic Generation of Multiple Pronunciations for Proper Nouns," Technical Report, Texas Instruments Inc., October 1997.
 - [16] F. Rosenblatt, "The Perceptron: A Perceiving and Recognizing Automaton", Cornell Aeronautical Laboratory Report 85-460-1, 1957.
 - [17] T. Kohonen, "Automatic Formation of Topological Maps in a Self-organizing System", E. Oja and O. Simula (Editors), *Proceedings of the 2nd Scandinavian Conference on Image Analysis*, pp. 214-220, 1981.
 - [18] J.L. Elman and D. Zipser, "Learning the Hidden Structure of Speech", ICS Report 8701, University of California at San Diego, 1987.
 - [19] A. Waibel, H. Sawai and K. Shikano, "Modularity and Scaling in Large Phonemic Time-delay Neural Networks", *IEEE Transactions on Acoustics, Speech and Signal Processing*, Vol. 30, pp. 1888-1898, December 1989.
 - [20] A.P. Dempster, N.M. Laird and D.B. Rubin, "Maximum Likelihood Estimation from Incomplete Data," *Journal of the Royal Statistical Society*, Vol. 39, No. 1, pp. 1-38, 1977.
 - [21] N. Deshmukh and J. Picone, "Automated Generation of N-Best Proper Noun Pronunciations", Technical Report, Texas Instruments Inc., August 1996.
 - [22] M.L. Minsky and S. Papert, *Perceptrons*, MIT Press, Cambridge, MA, 1969.

- [23] J.J. Hopfield, "Neural Networks and Physical Systems with Emergent Collective Computational Abilities", *Proceedings of the National Academy of Sciences USA*, Vol. 79, pp. 2554-2558, 1982.
- [24] D.E. Rumelhart, G.E. Hinton and R.J. Williams, "Learning Internal Representation by Error Propagation," D.E. Rumelhart and J.L. McClelland (Editors), *Parallel Distributed Processing: Explorations in The Microstructures of Cognition, Vol. 1: Foundations*, MIT Press, Cambridge, MA, pp. 318-362, 1986.
- [25] R. Rosenfeld, "A Hybrid Approach to Adaptive Statistical Language Modeling," *Proceedings ARPA Workshop on Human Language Technology*, pp. 76-81, Plainsboro, NJ, March 1994.
- [26] T. Robinson, M. Hochberg, and S. Renals, "IPA: Improved Phone Modeling With Recurrent Neural Networks," *Proceedings IEEE International Conference on Acoustics, Speech, and Signal Processing*, pp. I-37-I-40, Adelaide, Australia, April 1994.
- [27] A. Waibel, T. Hanazawa, G. Hinton, K. Shikano, and K. Lang, "Phoneme Recognition Using Time-Delay Neural Networks," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, Vol. 37, no. 3, pp. 328-339, March 1989.
- [28] S. Kirkpatrick, C.D. Gellatt and M.P. Vecchi, "Optimization by Simulated Annealing," *Science*, Vol. 220, pp. 671-680, 1983.
- [29] D.A. Ackley, G.E. Hinton and T.J. Sejnowski, "A Learning Algorithm for Boltzmann Machines," *Cognitive Science*, Vol. 9, pp. 147-169, 1985.
- [30] W.M.P. Daelemans and A.P.J. van den Bosch, "Language-Independent Data-Oriented Grapheme-to-Phoneme Conversion," J.P.H. van Santen, R.W. Sproat, J.P. Olive and J. Hirschberg (Editors), *Progress in Speech Synthesis*, pp. 77-89, Springer-Verlag, New York, NY, 1996.
- [31] S.J. Young, "A Review of Large-Vocabulary Continuous-Speech Recognition," *IEEE Signal processing Magazine*, Vol. 13, No. 5, pp. 45-57, September 1996.
- [32] H. Ney, U. Essen and R. Kneser, "On Structuring Probabilistic Dependencies in Stochastic Language Modeling," *Computer Speech and Language*, Vol. 8, No. 1, pp. 1-38, 1994.
- [33] S.M. Katz, "Estimation of Probabilities from Sparse Data for the Language Model Component of a Speech Recognizer," *IEEE Transactions on Acoustics, Speech and Signal Processing*, Vol. 35, No. 3, pp. 400-401, 1987.

- [34] B. Wheatley and J. Picone, "Integrating Speech Technologies For Medical Applications," presented at *the Medical Applications of Voice Response Technology Conference*, Pittsburgh, PA, December 1989.
- [35] A.J. Vitale, "Application-Driven Technology: Automated Customer Name and Address," *Proceedings of the AVIOS*, pp. 33-40, March 1989.
- [36] R. Carlson, B. Granstrom and A. Lindstrom, "Predicting Name Pronunciations for a Reverse Directory Assistance," *Proceedings of the 1st European Conference on Speech Communication and Technology*, Vol. 1, pp. 113-116, 1989.
- [37] J. Picone, B.J. Wheatley and J. McDaniel, "On the Intelligibility of Text-To-Speech Synthesis of Surnames," *Texas Instruments Technical Report No. CSC-TR-91-002*, pp. 1-34, Texas Instruments Inc., Dallas, TX, March 13, 1991.
- [38] D. Conroy, A.J. Vitale and D.H. Klatt, *DECTalk DTC03 Text-to-Speech System Owner's Manual, EK-DTC-03-OM-001*, Digital Equipment Corporation, 1986.
- [39] M.F. Spiegel and M.J. Macchi, "Synthesis of Names by a Demisyllable-Based Speech Synthesizer (Orator)," *AVIOS Journal*, Vol. 7, Special RHOC/RBOC Issue, 1990.
- [40] C.H. Coker, K.W. Church and M.Y. Lieberman, "Morphology and Rhyming: Two Powerful Alternatives to Letter-to-Sound Rules for Speech Synthesis," *Conference on Speech Synthesis*, European Speech Communication Association, 1990.
- [41] A.R. Golding and P.S. Rosenbloom, "The Evaluation of Anapron: A Case Study in Evaluating a Case-based System," *Working Notes of the AAI-94 Workshop on Case-Based Reasoning*, pp. 84-90, Seattle, WA, 1994.
- [42] H.M. Meng, S. Seneff, and V.W. Zue, "Phonological Parsing for Reversible Letter-to-Sound/Sound-to-Letter Generation," *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, pp. II-1-II-4, Adelaide, Australia, April 1994.
- [43] G. Hinton and J. Anderson (Eds.), *Parallel Models of Associative Memory*, Erlbaum Associates, NJ, 1981.
- [44] T.J. Sejnowski and C.R. Rosenberg, "NETtalk: A Parallel Network That Learns To Read Aloud," *Technical Report JHU/EECS-86/01*, John Hopkins University, Baltimore, MD, 1986.
- [45] B.J. Wheatley and J. Picone, "Voice Recognition of proper nouns Using

- Text-Derived Recognition Models,” *US Patent No. 5212730*, May 18, 1993.
- [46] G. Hinton and T. Sejnowski, “Optimal Perceptual Inference,” *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 448-453, Washington D.C., June 1986.
- [47] D.R. Calvert, *Descriptive Phonetics*, 2nd Edition, Thieme Inc., New York, New York, USA, 1986.
- [48] M.D. Riley, “A Statistical Model for Generating Pronunciation Networks,” *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, S11.1, pp. 737-740, Toronto, Canada, May 1991.
- [49] A.R. Golding and P.S. Rosenbloom, “A Comparison of Anapron with Seven Other Name-Pronunciation Systems,” *Journal of the American Voice Input/Output Society*, Vol. 14, pp. 1-21, August 1993.
- [50] S.D. Basson, D. Yashchin, K. Silverman and A. Kalyanswamy, “Assessing the Acceptability of Automated Customer Name and Address: A Rigorous Comparison of Text-to-Speech Synthesizers,” *Proceedings of AVIOS*, 1991.
- [51] D.H. Klatt, “Review of Text-to-Speech Conversion for English,” *Journal of the Acoustical Society of America*, Vol. 82, pp. 737-793, 1987.
- [52] K.W. Church, “Stress Assignment in Letter-to-Sound Rules for Speech Synthesis,” *Proceedings of the 23rd Meeting of the Association of Computational Linguistics*, pp. 246-253, 1985.
- [53] R.L. Venezky, *A Study of English Spelling-to-Sound Correspondence on Historical Principles*, Ann Arbor Press, Ann Arbor, MI, 1965.
- [54] N. Chomsky and M. Halle, *Sound Pattern of English*, Harper and Row, New York, NY, 1968.
- [55] L. Henderson, *Orthography and Word Recognition in Reading*, Academic Press, New York, NY, 1982.
- [56] A. Wijk, *Alphabets for English*, W. Haas (Ed.), Manchester University Press, Manchester, U.K., 1969.
- [57] W.A. Ainsworth, “A System for Converting English Text into Speech,” *IEEE Transactions on Audio Electroacoustics*, Vol. 21, pp. 288-290, 1973.

- [58] R. Carlson and B. Granstrom, "A Text-to-speech System Based Entirely on Rules," *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 686-688, 1976.
- [59] M. Halle and S.J. Keyser, *English Stress: Its Form, its Growth and its Role in Verse*, Harper and Row, New York, NY, 1971.
- [60] S. Hunnicutt, "Phonological Rules for a Text-to-Speech System," *American Journal of Computational Linguistics*, Microfiche 57, pp. 1-72, 1976.
- [61] K. Hill and L. Nessly, "Review of Sound Patterns in English," *Linguistics*, Vol. 106, pp. 57-101, 1973.
- [62] J. Bernstein and L. Nessly, "Performance Comparison of Component Algorithms for the Phonemicization of Orthography," *Proceedings of the 19th Annual Meeting of the Association of Computational Linguistics*, pp. 19-22, Pasadena, CA, 1981.
- [63] S. Hunnicutt, "Grapheme-to-Phoneme Rules, A Review," *QPSR 2-3*, Speech Transmission Laboratory, Royal Institute of Technology, pp. 38-60, Stockholm, Sweden, 1980.
- [64] H. Elovitz, R. Johnson, A. McHugh and J. Shore, "Letter-to-Sound Rules for Automatic Translation of English Text to Phonetics," *IEEE Transactions on Acoustics, speech and Signal Processing*, Vol. 24, pp. 446-459, 1976.
- [65] S. Hertz, "From Text to Speech with SRS," *Journal of the Acoustical Society of America*, Vol. 72, pp. 1155-1170, 1982.
- [66] A.W.F. Huggins, R.A. Houde and E. Colwell, "Vidvox Human factors Investigation," *Technical Report No. 6187*, pp. 15, Bolt Beranek and Newman Inc., Cambridge, MA, 1986.
- [67] L. Bloomfield, *Language*, Henry Holt, New York, NY, 1933.
- [68] F.W. Fischer, D. Shankweiler and I.Y. Liberman, "Spelling Proficiency and Sensitivity to Word Structure," *Journal of Memory Languages*, Vol. 24, pp. 423-441, 1985.
- [69] F.F. Lee, "Reading Machines: From Text to Speech," *IEEE Transactions on Audio Electroacoustics*, Vol. 17, pp. 275-282, 1969.
- [70] J. Allen, S. Hunnicutt, R. Carlson and B. Granstrom, "MITalk-79: The MIT Text-to-Speech System," *Journal of the Acoustical Society of America*, Supplement

- 1, Vol. 65, S130, 1979.
- [71] H. Kucera and W.N. Francis, *Computational Analysis of Present-day American English*, Brown University Press, Providence, RI, 1967.
- [72] J. Allen, S. Hunnicutt and D.H. Klatt, *From Text to speech: The MITalk System*, Cambridge University Press, Cambridge, U.K., 1987.
- [73] C.H. Coker, "A Dictionary-Intensive Letter-to-Sound Program," *Journal of the Acoustical Society of America*, Supplement 1, Vol. 78, S7, 1985.
- [74] G.F. Groner, J. Bernstein, E. Ingber, J. Pearlman and T. Toal, "A Real-Time Text-to-Speech Converter," *Speech Technology*, Vol. 1, pp. 73-76, 1982.
- [75] B.J. Malsheen, "Recent Developments in Text-to-Speech Research," *Proceedings of the National Communications Forum*, Vol. 43, No. 1, pp. 1104-1108, October 1989.
- [76] C.E. Wright, M.J. Altom and J.P. Olive, "Diagnostic Evaluation of a Synthesizer's Acoustic Inventory," *Journal of the Acoustical Society of America*, Supplement 1, Vol. 79, S25, 1986.
- [77] J.E. Davoust and M.F. Spiegel, "How Well Do State-of-the-Art Speech Synthesizers Pronounce Names," *Technical Memorandum TM-TSV-016139*, Bell Communications Research, Morristown, NJ, February 1990.
- [78] K.W. Church, "Stress Assignment in Letter-to-Sound Rules for Speech Synthesis," *Proceedings of the 23rd Meeting of the Association of Computational Linguistics*, pp. 246-253, 1985.
- [79] A.J. Vitale, "An Algorithm for High Accuracy Name Pronunciation by Parametric Speech Synthesizer," *Journal of Computational Linguistics*, Vol. 17(3), 1991.
- [80] A.R. Golding, "Pronouncing Names by a Combination of Rule-Based and Case-Based Reasoning," *Ph.D. Thesis*, Stanford University, Pasadena, CA, 1991.
- [81] A.R. Golding and P.S. Rosenbloom, "Improving Rule-Based Systems Through Case-Based Reasoning," *Proceedings of AAI*, Anaheim, Canada, 1991.
- [82] R.J. Glushko, "Principles for Pronouncing Print: The Psychology of Phonography," *Interactive Processes in Reading*, A.M. Lesgold and C.A. Perfetti (Editors), Earlbaum, Hillsdale, NJ, 1981.
- [83] M.J. Dedina and H.C. Nusbaum, "Pronounce: A Program for Pronunciation by

- Analogy,” *Speech Research Laboratory Progress Report 12*, Indiana University, Bloomington, IN, pp. 335-348, 1986.
- [84] F. Yvon, “Grapheme to Phoneme Conversion Using Multiple Unbounded Overlapping Chunks,” *Proceedings of the Conference on New Methods in Natural Language Processing*, pp. 218-228, Ankara, Turkey, 1996.
- [85] R.I. Damper and J.F.G. Eastmond, “Pronunciation by Analogy: Impact of Implementational Choices on Performance,” *Language and Speech*, Vol. 40, pp. 1-23, 1997.
- [86] M.J. Adamson and R.I. Damper, “A Recurrent Network That Learns to Pronounce English Text,” *Proceedings of the 4th International Conference on Spoken Language Processing*, Vol. 3, pp. 1704-1707, Philadelphia, PA, October 1996.
- [87] P.C. Bagshaw, “Phonetic Transcription by Analogy in Text-to-Speech Synthesis: Novel Word Pronunciation and Lexicon Compression,” *Computer Speech and Language*, Vol. 12, pp. 119-142, 1998.
- [88] S. Deligne, F. Yvon and F. Bimbot, “Variable-Length Sequence Matching for Phonetic Transcription Using Joint Multigrams,” *Proceedings of the 4th European Conference on Speech Communication and Technology*, Vol. 3, pp. 2243-2246, Madrid, Spain, 1995.
- [89] K.P.H. Sullivan and R.I. Damper, “Synthesis-by-Analogy: A Bilingual Investigation Using German and English,” *Proceedings of the 2nd International Conference on Spoken Language Processing*, Vol. 1, pp. 113-116, Banff, Canada, 1992.
- [90] J.M. Lucassen and R.L. Mercer, “An Information Theoretic Approach to the Automatic Determination of Phonemic Baseforms,” *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 42.5.1-42.5.4, 1984.
- [91] M.F. Tenorio, M.D. Tom and R.G. Schartz, “Adaptive Networks as a Model for Human Speech Development,” *Proceedings of the 2nd International Conference on Neural Networks*, Vol. 2, pp. 235-242, 1988.
- [92] D.H. Klatt and D.W. Shipman, “Letter-to-Phoneme Rules: A Semi-Automatic Discovery Procedure,” *Journal of the Acoustical Society of America*, Supplement 1, Vol. 72, S48, 1982.
- [93] K. Torkkola, “An Efficient Way to Learn English Grapheme-to-Phoneme Rules Automatically,” *Proceedings of the IEEE International Conference on Acoustics,*

- Speech and Signal Processing*, Vol. 2, pp. 199-202, 1993.
- [94] R.W.P. Luk and R.I. Damper, "A Novel Approach to Inferring Letter-Phoneme Correspondences," *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, Vol. 2, pp. 741-744, 1991.
- [95] R.W.P. Luk and R.I. Damper, "Inference of Letter-Phoneme Correspondences with Predefined Consonant and Vowel Patterns," *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, Vol. 2, pp. 203-206, 1993.
- [96] R.W.P. Luk and R.I. Damper, "Stochastic Phonographic Transduction for English," *Computer Speech and Language*, Vol. 10, pp. 133-153, 1996.
- [97] R.W.P. Luk and R.I. Damper, "Computational Complexity of a Fast Viterbi Decoding Algorithm for stochastic Letter-Phoneme Transduction," *IEEE Transactions on Speech and Audio Processing*, Vol. 6, No. 3, pp. 217-225, May 1998.
- [98] N. McCulloch, M. Bedworth and J.S. Bridle, "NETspeak — A Reimplementation of NETtalk," *Computer Speech and Language*, Vol. 2, pp. 289-301, 1987.
- [99] T.G. Diettrich, H. Hild and G. Bakiri, "A Comparative Study of ID3 and Back-Propagation for English Text-to-Speech Mapping," *Proceedings of the 7th International Machine Learning Workshop*, Austin, TX, 1990.
- [100] R. Sproat (Editor), *Multilingual Text-to-Speech Synthesis — The Bell Labs Approach*, Kluwer Academic Publishers, Netherlands, 1998.
- [101] A.W. Black, K. Lenzo and V. Pagel, "Issues in Building General Letter to Sound Rules," *Proceedings of the ESCA Speech Synthesis Workshop*, Jenolan Caves, Blue Mountains, Australia, 1998.
- [102] V. Pagel, K. Lenzo and A.W. Black, "Letter to Sound Rules for Accented Lexicon Compression," *Proceedings of the International Conference on Spoken Language Processing*, Sydney, Australia, October 1998.
- [103] L.R. Bahl, F. Jelinek and R.L. Mercer, "A Statistical Approach to Continuous Speech Recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1983.
- [104] F. Jelinek, "Up From Trigrams! The Struggle for Improved Language Models," the *Proceedings of the 2nd European Conference on Speech Communication and Technology (Eurospeech)*, pp. 1037-1040, Genova, Italy, September 1991.

- [105] X.D. Huang, F. Alleva, H.W. Hon, M.Y. Hwang, K.F. Lee and R. Rosenfeld, "The SPHINX-II Speech Recognition System: An Overview," *Computer, Speech and Language*, 1992.
- [106] R. Rosenfeld and X.D. Huang, "Improvements in Stochastic Language Modeling," *Proceedings of the DARPA Speech and Natural Language Workshop*, February 1992.
- [107] R. Rosenfeld, "A Hybrid Approach to Adaptive Statistical Language Modeling," *Proceedings DARPA Human Language Technology Workshop*, pp. 76-81, March 1994.
- [108] R. Rosenfeld, "Adaptive Statistical Language Modeling: A Maximum Entropy Approach," Ph.D. Thesis Proposal, Carnegie Mellon University, September 1992.
- [109] R. Lau, R. Rosenfeld and S. Roukos, "Trigger-Based Language Models: A Maximum Entropy Approach," *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, Vol. 2, pp. 45-48, April 1993.
- [110] J. Kupiec, "Probabilistic Models of Short and Long Distance Word Dependencies in Running Text," *Proceedings of the ARPA Workshop on Speech and Natural Language*, pp. 290-295, February 1989.
- [111] F. Jelinek, B. Merialdo, S. Roukos and M. Strauss, "A Dynamic LM for Speech Recognition," *Proceedings of the ARPA Workshop on Speech and Natural Language*, pp. 293-295, 1991.
- [112] R. Kuhn and R. de Mori, "A Cache Based Natural Language Model for Speech Recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 14, pp. 570-583, 1992.
- [113] M. Jardino, "Multilingual Stochastic N-gram Class Language Models," *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, Vol. 1, pp. 161-163, May 1996.
- [114] L. Bahl, P.F. Brown, P.V. de Souza and R.L. Mercer, "A Tree-Based Statistical Language Model for Natural Language Speech Recognition," *IEEE Transactions on Acoustics, Speech and Signal Processing*, Vol. 37, No. 7, pp. 1001-1008, 1989.
- [115] R. Kneser and V. Steinbiss, "On the Dynamic Adaptation of Stochastic LM," *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, Vol. 2, pp. 586-589, April 1993.

- [116] R. Iyer, M. Ostendorf and J.R. Rohlicek, "An Improved Language Model Using a Mixture of Markov Components," *Proceedings of the DARPA Human Language Technology Workshop*, pp. 82-86, March 1994.
- [117] R. Iyer, "Language Modeling with Sentence-Level Mixtures," M.S. Thesis, Boston University, 1994.
- [118] H. Ney, "Dynamic Programming Speech Recognition Using A Context-Free Grammar," *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 321-324, April 1987.
- [119] C. Hemphill and J. Picone, "Robust Speech Recognition in a Unification Grammar Framework," *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 723-726, May 1989.
- [120] J.K. Baker, "Stochastic Modeling as a Means of Automatic Speech Recognition," Ph.D. Thesis, Carnegie Mellon University, 1975.
- [121] L.R. Bahl, J.K. Baker, P.S. Cohen, A.G. Cole, F. Jelinek, B.L. Lewis and R.L. Mercer, "Automatic Recognition of Continuously Spoken Sentences from A Finite State Grammar," *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 418-421, April 1978.
- [122] M.D. Riley, "A Statistical Model for Generating Pronunciation Networks," *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, Vol. 2, pp. 737-740, Toronto, Canada, 1991.
- [123] M.D. Riley and A. Ljolje, "Automatic Generation of Detailed Pronunciation Lexicons," *Automatic Speech and Speaker Recognition: Advanced Topics*, C-H. Lee, F.K. Soong and K.K. Paliwal (Editors), Kluwer Academic Press, Netherlands, 1996.
- [124] G. Tajchman, E. Fosler and D. Jurafsky, "Building Multiple Pronunciation models for Novel words Using Exploratory Computational Phonology," *Proceedings of the 4th European Conference on Speech Communication and Technology*, pp. 2247-2250, Madrid, Spain, 1995.
- [125] M. Weintraub, E. Fosler, C. Galles, Y-H. Kao, S. Khudanpur, M. Saraclar and S. Wegmann, "Automatic Learning of Word Pronunciations from Data," *Proceedings of the DARPA LVCSR Summer Research Workshop*, Johns Hopkins University, 1996.
- [126] T.J. Sejnowski and C.R. Rosenberg, "Parallel Networks That Learn to Pronounce English Text," *Complex Systems*, Vol. 1, pp. 145-168. 1987.

- [127] L. Rabiner and B-H. Juang, *Fundamentals of Speech Recognition*, Prentice Hall, Englewood Cliffs, NJ, 1993.
- [128] M.P. Marcus, B. Santorini and M.A. Marcinkiewicz, "Building a Large Annotated Corpus of English: The Penn Treebank," *Computational Linguistics*, Vol. 19, pp. 313-330, 1993.
- [129] M. Schmidt, S. Fitt, C. Scott and M. Jack, "Phonetic Transcription Standards for European Names (ONOMASTICA)," *Proceedings of the 3rd European Conference on Speech Communication and Technology*, Vol. 1, pp. 279-282, Berlin, Germany, 1993.
- [130] N. Deshmukh and J. Picone, "Automatic Generation of Proper Noun Pronunciations," http://www.isip.msstate.edu/projects/nbest_pronunciations/, Institute for Signal and Information Processing, Mississippi State University, MS, 1998.
- [131] E.C. Smith, *American Surnames*, Genealogical Publishing Co. Inc., Baltimore, MD, 1986.
- [132] R.A. Cole, M. Fanty and K. Roginski, "A Telephone Speech Database of Spelled and Spoken Names," *Proceedings of the International Conference on Spoken Language Processing*, pp. 891-893, Banff, Alberta, Canada, October 1992.
- [133] S.G.C. Lawrence and G. Kaye, "Alignment of Phonemes with Their Corresponding Orthography," *Computer Speech and Language*, Vol. 1, pp. 153-165, 1986.
- [134] B.J. Malsheen and M. Amador-Hernandez, "The Interrelationship of Intelligibility and Naturalness in Text-to-Speech," *Proceedings of the International Conference on Spoken Language Processing*, pp. 333-336, November 1990.
- [135] K. Silverman, S. Basson and S. Levas, "Evaluating Synthesizer Performance: Is Segmental Intelligibility Enough?," *Proceedings of the International Conference on Spoken Language Processing*, pp. 981-984, November 1990.
- [136] N. Deshmukh, M. Weber and J. Picone, "Automated Generation of N-Best Pronunciations of Proper Nouns," *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, Vol. 1, pp. 283-286, Atlanta, GA, May 1996.
- [137] N. Deshmukh, A. Le, J. Ngan, J. Hamaker and J. Picone, "An Advanced System to Generate Multiple Pronunciations of Proper Nouns," *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, pp.

1467-1470, Munich, Germany, April 1997.

- [138] J. Ngan, "Information Theory Based Decision Trees for Data Classification," http://www.isip.msstate.edu/publications/seminars/masters_oral/1998/decision_tree_c4/, Master of Science Special Project Presentation, Institute for Signal and Information Processing, Mississippi State University, December 1998.
- [139] A. Le, "Bayesian Decision Tree for Classification of Nonlinear Signal Processing Problems," http://www.isip.msstate.edu/publications/seminars/masters_oral/1998/decision_tree_bayes/, Master of Science Special Project Presentation, Institute for Signal and Information Processing, Mississippi State University, November 1998.
- [140] J. Ngan, A. Ganapathiraju and J. Picone, "Improved Surname Pronunciations Using Decision Trees," *Proceedings of the International Conference on Spoken Language Processing*, Vol. 7, pp. 3285-3288, Sydney, Australia, November 1998.
- [141] M. Schuster, Personal communication.
- [142] N. Deshmukh, A. Ganapathiraju, J. Hamaker and J. Picone, "An Efficient Public Domain LVCSR Decoder," *Proceedings of the Hub-5 Conversational Speech Recognition (LVCSR) Workshop*, Linthicum Heights, Maryland, USA, September 1998.
- [143] N. Deshmukh, A. Ganapathiraju, J. Hamaker and J. Picone, "Large Vocabulary Conversational Speech Recognition," <http://www.isip.msstate.edu/projects/speech/>, Mississippi State University, 1999.