Advances in Automatic Generation of Multiple Pronunciations for Proper Nouns

prepared for:

Speech Research Group
Personal Systems Laboratory
Texas Instruments, Inc.
PO Box 655474, MS 238
Dallas, Texas 75265

by:

N. Deshmukh, A. Le and J. Picone
Institute for Signal and Information Processing
Department of Electrical and Computer Engineering
Mississippi State University
Box 9571
413 Simrall, Hardy Rd.
Mississippi State, Mississippi 39762
Tel: 601-325-3149
Fax: 601-325-3149
email: {le, deshmukh,picone}@isip.msstate.edu

EXECUTIVE SUMMARY

A persistent challenge in achieving high-accuracy human-computer speech interactions is the problem of proper noun recognition. For many voice-based interfaces such as those applied to directory assistance or medical records databases, accurate recognition of proper nouns (such as customer name, or the name of the doctor / patient) is of critical importance. In order to achieve this, the system requires the capability of generating accurate pronunciation models for such words. This is a very challenging problem due to the lack of any obvious letter-to-sound mapping rules, and multiple pronunciations derived from a wide variety of ethnolinguistic factors for these proper nouns. Traditional rule-based systems have serious shortcomings of range and generalization. Thus it is imperative to develop a mechanism for automatically generating multiple pronunciations for proper nouns.

In this project, the Institute for Signal and Information Processing (ISIP) at Mississippi State University has endeavored to implement a classification system that captures the statistics of pronunciation units (phones) as a function of the orthographic spelling of the proper noun. In the first year of this project, ISIP implemented a hybrid neural networks algorithm capable of discovering some underlying structure in this letter n-tuples — to — phones domain to provide Texas Instruments (TI) with a software system capable of converting any proper noun to a list of its most plausible pronunciations. This research effort also involved the development of a representative database containing approximately 18000 surnames and their possible pronunciations; hand-transcribed and automatically aligned with the spellings.

This year was devoted to the analysis of the basic neural network system and in exploring avenues to enhance its performance. Two different directions were pursued. In the first, the existing neural network system was modified by adding new features such as the ability to support topologies with multiple hidden layers of neurons. Different topologies such as a multilayered version of the old system, a non-stochastic multilayered perceptrons and learning vector quantizer were implemented. A hierarchical system that performed two-level classification — first dividing the data into phone classes (such as vowels, diphthongs, consonants and nulls) and then classifying to the individual phone under each cluster — was also tried out. The training algorithm was modified to improve the rate of training as well as the performance of the trained system. The second approach involved implementation of an elementary decision-tree (AT) based classifier that assigns phones to the proper noun spellings based on a series of context-dependent binary decisions. Both systems were trained on the pronunciation dictionary database and extensively evaluated using the paradigms developed in the first year of the project.

The deliverables for this project include the complete software and binaries for training the system and generating the N-best pronunciations. The software adheres to the standards of C++ object-oriented data-driven applications at ISIP. An updated Tcl-Tk graphical user interface (GUI) demo allows the user to select the algorithm type (ANN or DT) and set the corresponding system parameters, and outputs an ordered list of the likely pronunciations along with a phonetic network that can generate them for an input name. The pronunciations database, as well as all tools, software and publications developed in this project are placed into the public domain (*http://www.isip.msstate.edu/resources/technology/projects/1997/nbest_pronunciations/*) and supported by ISIP.

TABLE OF CONTENTS

1. ABSTRACT

The ability to accurately model word pronunciations is crucial to the effectiveness of a voice-based human-computer interface. For speech recognition and/or synthesis systems that deal with customer-oriented applications such as medical databases and telephone directory assistance, the generation of correct pronunciations for proper nouns is even more critical. However, this turns out to be a very complicated problem as proper nouns are often found to depart from typical letter-to-sound conversion rules followed by regular words. Moreover, many proper nouns have multiple valid pronunciations that evolve as a product of various sociolinguistic phenomena, and the system needs to generate accurate pronunciation networks for correct identification. Traditional rule-based systems are incapable of comprehensively dealing with this problem.

In the first year of this project, we developed a stochastic neural network paradigm that automatically generates a rank-ordered list of the various possible pronunciations from textual spellings of proper nouns and evaluated its performance. The second year was expended in enhancing the performance of the neural network system, as well as exploring new technology such as decision trees. This document describes the significant changes made to the neural network system as well as the implementation of a decision-tree based system. Both systems have been trained and evaluated on the same database of 18494 surnames and 24000 pronunciations. The performance of the neural network system improved from 63.95% error to 52.34%; while the decision tree system recorded 47.13% error in generating pronunciations of unseen data.

2. INTRODUCTION

A critical aspect of voice-driven interfaces is their ability to perform accurate recognition of proper nouns. For instance, In many applications related to medicine, the ability to recognize a physician’s or patient’s name is crucial in providing a usable interface. A comparable problem involving company names and product names exists in voice interfaces for advanced telecommunications services. It is also well-known that a majority of errors in a continuous speech recognition system consist of proper nouns, primarily due to a lack of good acoustic models for such out-of-vocabulary words.

Recognition of proper nouns requires an ability to generate accurate pronunciation networks. This problem is very challenging because a large percentage of proper nouns, such as surnames, have no obvious letter-to-sound mapping rules that can be used to generate the pronunciations. Instead, the pronunciations seem to be a product of numerous sociolinguistic factors. While a traditional recourse to generate pronunciation models for proper nouns has been to handcraft massive rule-based systems and determine the best pronunciation, there is certainly need for a data-driven automated system capable of performing such a task. We have explored two statistical classification technologies — neural networks and decision trees — to tackle this problem.

2.1. Historical Background

The principal idea behind this work dates back to the late 1980s, when a feasibility study was conducted at Texas Instruments to evaluate the accuracy of text-to-speech (TTS) systems on pronouncing proper nouns [1] for an application in interfacing with a medical records database

[2]. It was found that the ability to recognize a physician's or patient's name was crucial in providing a usable interface, and such name recognition was a vital step in transforming medical record access from keyboard input to voice input.

A major drawback of such a system was that it required an extensive set of handwritten letter-to-sound rules which made the system cumbersome and expensive to develop and maintain. Moreover, such a rule-based system was constrained by its ability to generate only one pronunciation for a given proper noun, and failed to generalize when presented with names not covered by the rule set. For instance, a commercial product called DECtalk [3, 4] was developed to convert unrestricted English text into speech, using a set of phonological rules and by handling exceptions with a lookup table. This method was found to be highly labor-intensive and very limited in scope. Other commercial systems such as Bellcore's Orator [5] and Bell Labs' TTS [6] and Mitsubishi's Anapron [7] follow variations on the same concepts of rule-based and/or dictionary-based lookups.

Alternative approaches have emerged since then that employ statistical techniques to model the stochastic distribution of pronunciations with respect to letters of the spelling of the noun. In many cases this distribution is obviously nonlinear, and sophisticated techniques such as Hidden Markov Models (HMMs) [8] and artificial neural networks (ANNs) [9, 10] have been applied to this problem with varying success.

ANNs are connectionist systems where the knowledge about the data is distributed over multiple processing units and the net exchange of information between these units. 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 ideally suited for such applications of letter-to-phone conversion.

Such a multilayered neural network model called NETtalk [11] demonstrated that a relatively small network can capture most of the significant regularities of typical English pronunciations, as well as absorb a fair number of irregularities. It also had the advantage of being language independent 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.

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 DTs to automatically generate detailed phonetic pronunciation networks from a coarse phonemic transcription [12] has been developed at AT&T. DT-based systems are also capable of generating more than one pronunciations. However, no system has been designed that can effectively model the peculiarities of proper noun pronunciations to generate multiple pronunciations.

2.2. Baseline Neural Network System

The concept of ANNs originates from the notion that complex computing or classification operations can be implemented by massive integration of individual computing units, each of which performs an elementary computation. An advantage of using ANNs for feature extraction and classification of patterns is their capability to capture the inherent functionality of the data without any *a priori* statistical characterization or parameterization. The network classifies the large input

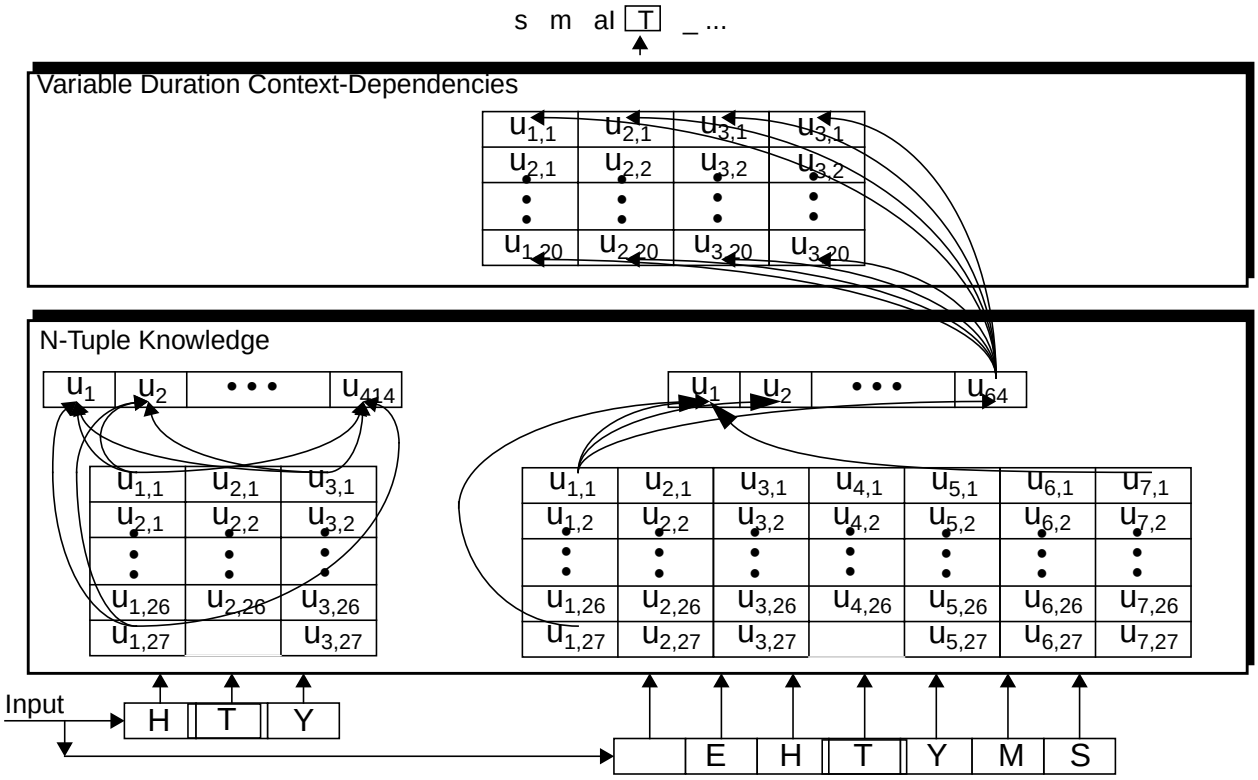

Figure 1. An overview of the neural network architecture that performs letter to sound conversion. In this system, there are three layers: a layer that converts letters to binary-valued inputs, a layer that converts n-tuples of letters into sounds, and a layer that applies a mixture of short-term and long-term relationships.

vectors into small output feature vectors representative of the input patterns. Often, such features constitute the internal activation patterns and not the output [13, 14] and the output classes represent certain associations of such features.

The pronunciation generation task is one of nonlinear classification of the letter n-tuples into different classes which correspond to various phonemes due to the nature of the data. Theoretically speaking, the stochastic neural network system has the ability to accurately solve such a problem.

In the first year of this project, we implemented an algorithm to automatically derive recognition models from the text-only spellings of the proper noun (rather than voice data containing a spelling or nominal pronunciation of the name). This system was proposed as a part of a technique for voice recognition of proper nouns using text-derived recognition models [15]. It relies upon a hybrid stochastic neural network [16] that combines the principles of multilayered feedforward networks [17, 18] and Boltzmann machines [11, 19]. It takes as input only the text-based spelling of the proper noun and generates an ordered N-best list of pronunciations.

This network looked at each character in the spelling of the noun in context of its left and right neighbors and tries to map such n-tuples of letters / characters to the corresponding sounds. A shift register structure was used to buffer characters as they are input to the system one at a time. It modeled the n-tuples of letters using local and long-distance constraints [20] using shift registers of different context window sizes, short and long. This approach is similar to other

time-delay techniques that have become popular in speech recognition systems [21, 22].

The architecture was designed specifically for the problem of name pronunciation / recognition and was based on the following two design criteria —

1. Generally, a relatively small amount of contextual information will be sufficient to narrow the range of possible sound correspondences to a small set.
2. Choosing a correct sound from this set may require information occurring at more remote points in the name (such as the identification of the foreign language from which the name was drawn).

Figure 1 describes the basic structure of such a network. It consists of three principal components described as follows —

1. 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 whitespace and a few other special characters such as the apostrophe and the period (a complete set of input characters is listed in Appendix B.1).
2. a hidden layer 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.
3. an output layer that mixes 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 (an illustrated description of the full phoneme set is provided in Appendix B.2).

There are two subnetworks possible in our design of the architecture, in order to capture both the short-distance as well as the long-term contextual information in the textual spelling of the name. Figure 2 illustrates the particular case in which a 3-tuple is used for short-term context while a 7-tuple is buffered to capture the long-context statistics.

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.

2.3. Deficiencies of the Baseline System

Our pilot experiments initially conducted on some synthetically generated data and later on a subset of the pronunciations database were quite encouraging with a high degree of accuracy in both phoneme classification as well as generating multiple pronunciations. However, when the

system was trained on the full training data set consisting of 15000 surnames and approximately 20000 pronunciations, the network performance degraded catastrophically. We identified the following problems with the functioning of the neural network system —

1. While the neural network system performs reasonably well when trained on a subset of the training database and tested likewise, it completely fails to capture the statistics of the full set of training data — there is a major scaling problem.
2. The performance of the system depends critically on the number of hidden neurons, yet there is no intuitive or scientific procedure to choose the optimum value for this parameter, or for any other important training parameters such as the initial system temperature and the cooling rate.
3. The network system supports only a single hidden layer of neurons. Multiple hidden neuron layers can possibly capture higher order statistics of the input data.
4. Training the network is an extremely expensive and computation-intensive process — each training iteration for a network with 300 hidden units and on the full training data set typically required 10 to 12 hours on a Sun SparcStation20, the system required at least 75 to 100 training iterations for moderately reasonable performance.

We also noticed that a significant fraction of the errors were related to particular phonemes being consistently misidentified with some other confusibles. This indicated that either the data was such that it made the letter n-tuples corresponding to certain phonemes very highly confusible, or that the neural network did not succeed in capturing some key features of the training data pertaining to those phonemes.

2.4. Modifications in the Baseline System

The performance of the baseline system on the full data set indicated a need for enhancements that would allow the system to scale up its discriminatory powers, so that it would be able to classify a larger number of classes (phonemes) from a more complex set of data. We have endeavored to achieve this in two different approaches — by trying to augment the baseline system with higher discriminatory features (by changing the topology, modifying the training etc.), and by exploring and implementing novel classification techniques.

Specifically, the following modification were applied to the baseline neural network system —

1. ability to support a number of connected hidden layers of neurons for the hybrid network.
2. a two-level classifier that first classifies the data into one of four phoneme superclasses (vowels, diphthongs, consonants and nulls) and then reclassifies it into the phonemes under the particular superclass.
3. implementation of a non-stochastic multilayered perceptron to accurately identify the one best possible pronunciation.
4. implementation of a learning vector quantizer to try a different classification paradigm.

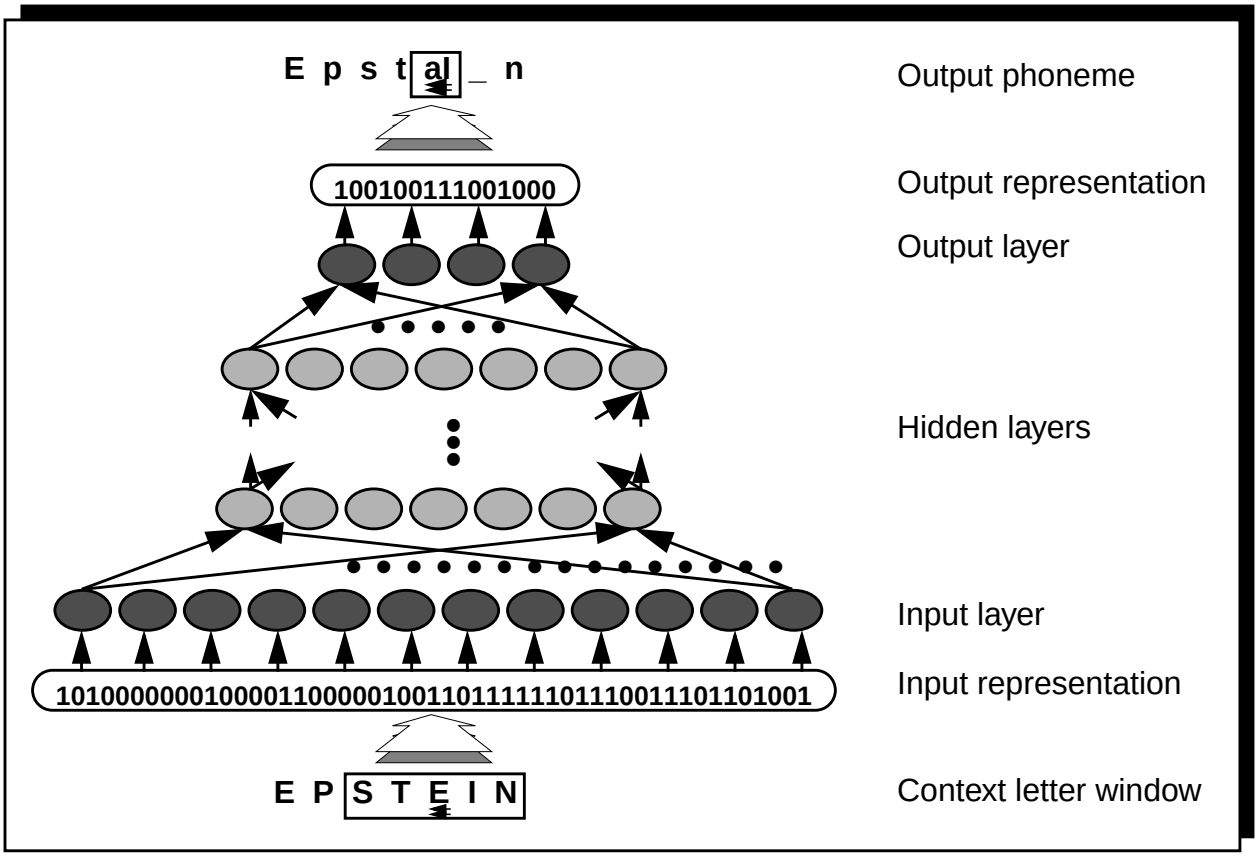

Figure 2. Topology for a multilayered feedforward network for pronunciation generation — the input layer connects to the bit-encoded letter n-tuples, the many hidden layers capture the letter-to-sound features in the input bit stream. The output is another bit stream which encodes the phoneme.

Both the multilayered perceptron and the learning vector quantizer are feedforward networks that are inherently 1-best i.e. generate only one ‘best’ answer. The training algorithm was also modified in accordance with each of the above enhancements. The basic backpropagation and simulated annealing algorithm used for the baseline system was also upgraded by using more efficient weight updating and more accurate activation functions.

Concurrently, we also implemented a simple decision-tree based classifier to identify the phonemes corresponding to input letter n-tuples.

2.5. Decision Trees Approach for Pronunciation Generation

Statistical decision trees have recently emerged as an important part of the state of the art in speech recognition technology, as they provide a flexible and completely data-driven tool for classification of complex, non-linearly separable data. Based on the response to a series of simple binary-valued questions, decision trees can efficiently and accurately generate classification clusters of highly complex shapes; and provide insights into the underlying phenomena and means of accurate prediction.

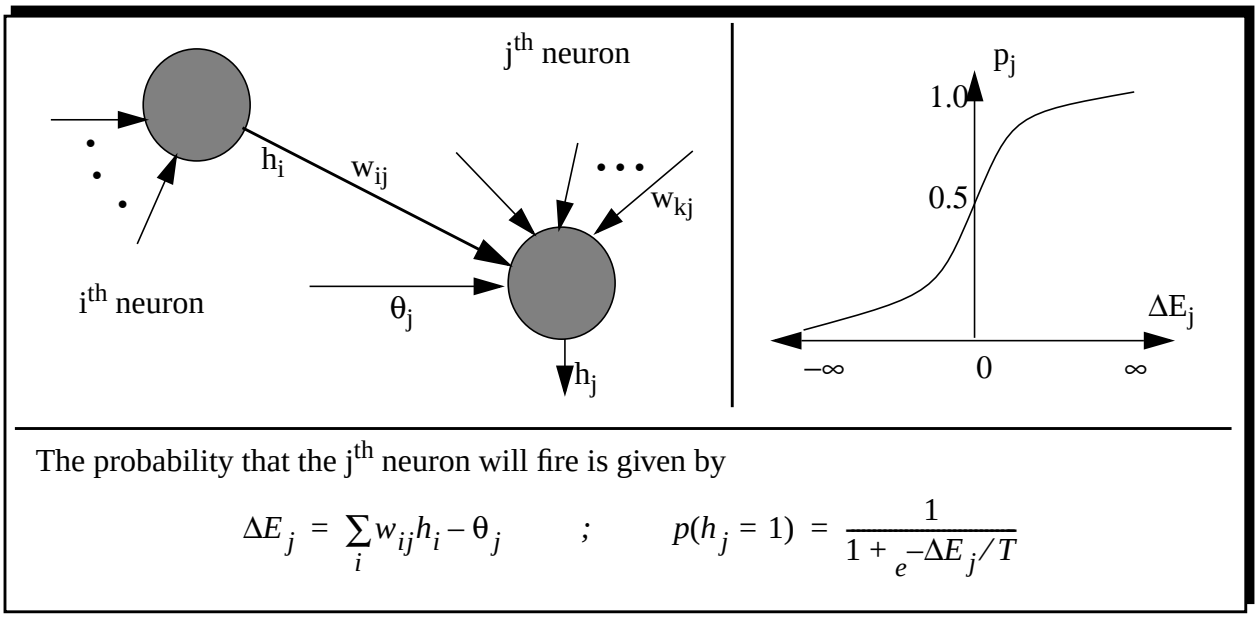

Figure 3. Typical neuron connections in a Boltzmann machine network, along with the Boltzmann distribution function that governs the activation probability of a neuron.

Decision trees have been successfully used for data exploration and classification in many diverse disciplines, such as astronomy [23], medical diagnostics [24], chemical analysis [25], pattern recognition [26] and detection [27]. Motivated by such versatile applicability of decision trees, we have developed a tree capable of classifying n-tuples of letters constituting the spelling of proper nouns into various phoneme classes to generate the corresponding pronunciations.

3. ENHANCEMENTS IN THE NEURAL NETWORK

As described in [28], in spite of the potential displayed in the pilot experiments, the performance of the baseline neural network system was not satisfactory at all when scaled up to the full data set. This highlighted several shortcomings of the architecture used that required remedial work. We implemented several variations in the topology of the neural network and the associated training algorithms to improve the pronunciation error rate.

3.1. Multiple Layers

Multilayered neural networks, in which the neurons can be viewed to be connected across different levels, form a significant class of pattern classification networks. In a multilayered feedforward network, the external inputs are fed to an initial ‘layer’ of neurons. The remaining cells constitute subsequent layers such that each successive layer receives as inputs the weighted outputs of the previous layer. The outputs of the final layer are the external outputs of the network. Such an architecture is also called a multilayered perceptron (MLP) [17, 18], and needs to be trained in a supervised fashion i.e. by exposing the network to the input patterns along with the corresponding desired outputs. An appropriately trained network can then reproduce the entire population of the target outputs as closely as possible in some sense.

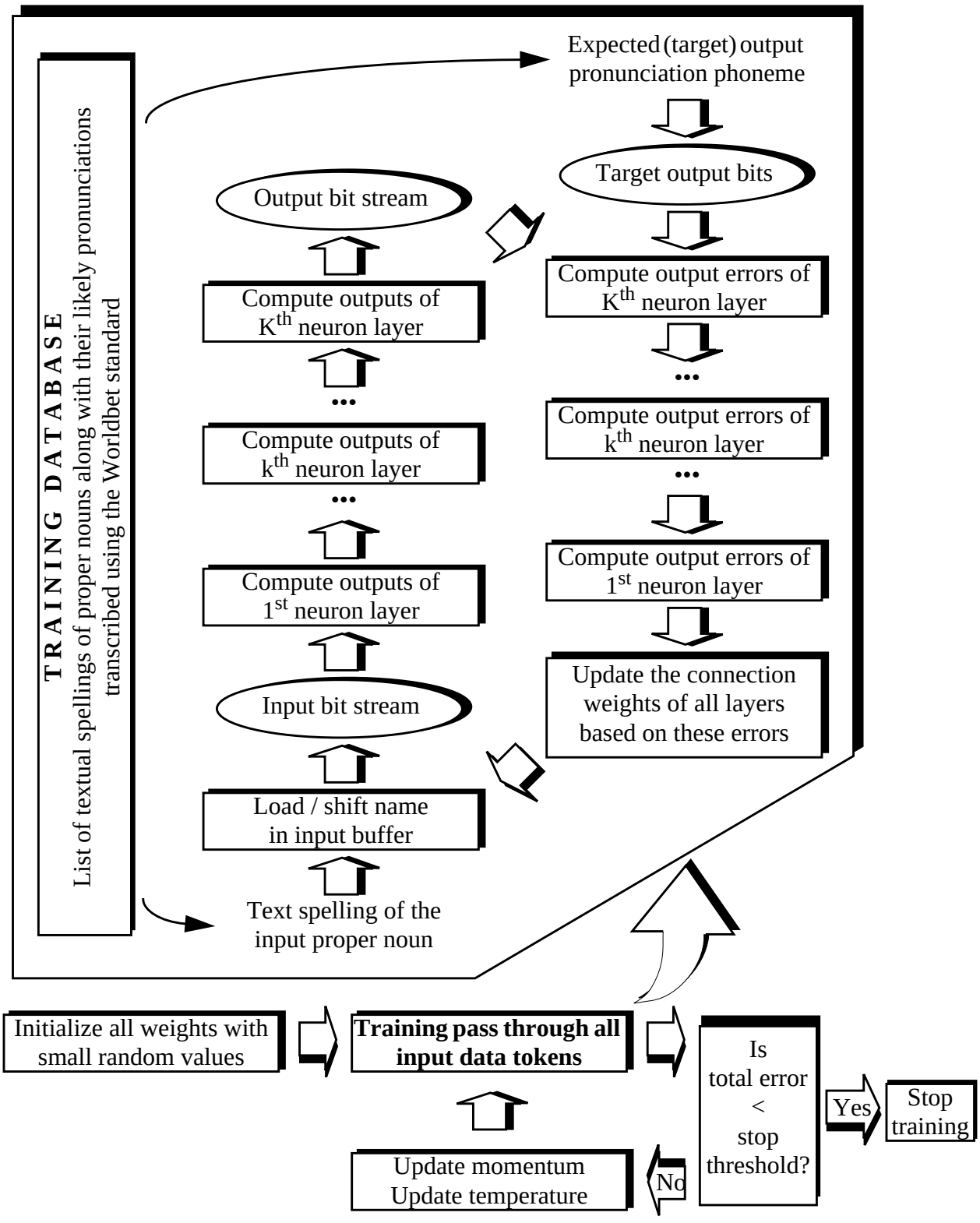

Figure 4. A schematic overview of the simulated annealing used to train the multilayered stochastic neural network. The backpropagation of error trough various layers of neurons during the training backward pass is displayed in detail in the inset.

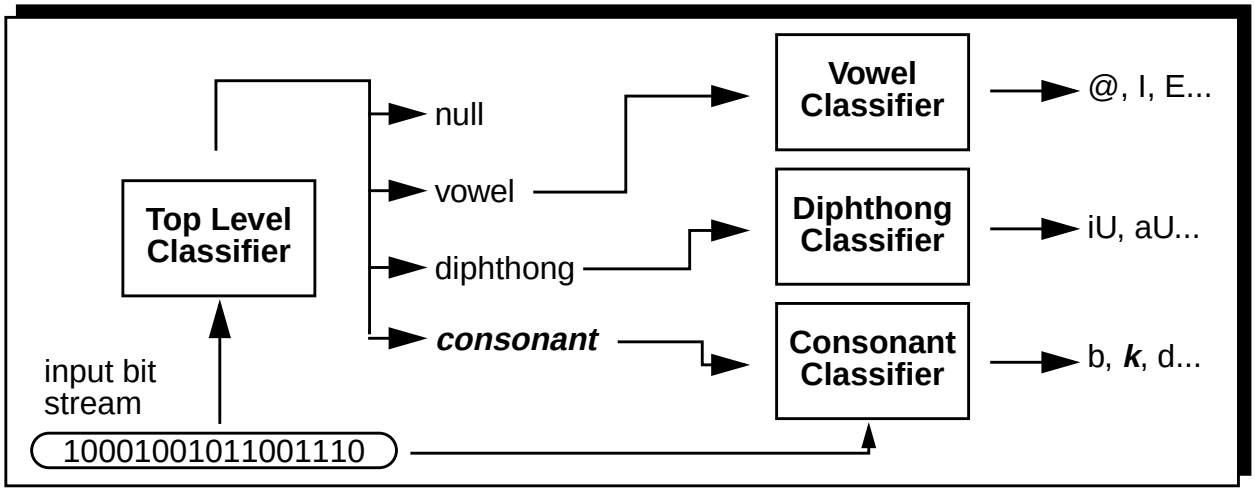

Figure 5. A schematic overview of the bilevel neural network classifier. The top level classifier decides whether the binary-encoded input n-tuple is a vowel, diphthong, consonant or a null phoneme. Accordingly, the same input is then fed to the corresponding low-level classifier to identify the actual phoneme. Here we illustrate a phoneme being identified as the consonant k.

While it has been theoretically proved that a multilayered perceptron with many hidden layers is equivalent to a feedforward network with a single hidden layer (with sufficiently large number of neurons), in practice it is often seen that with multiple layers the training is faster as smaller-sized layers are needed at each step [29]. Also, since the performance is a function of the hidden neurons, a change in topology also affects the classification to some extent.

We implemented a Boltzmann machine-type stochastic multilayered perceptron to provide the baseline system with the ability to capture the statistics of the data in a more efficient and effective manner. Figure 2 illustrates a typical neuron connection in such a network, and Figure 3 displays the stochastic activation function for an individual neuron. This network was trained using the standard training algorithm described in Figure 4, which incorporates elements of both backpropagation [30] and simulated annealing [31].

3.2. Hierarchical Bilevel Neural Network System

Our experience with the baseline neural network system amply demonstrated the complexity of the classification problem associated with the generation of phonemes from letter n-tuples. In order to improve the performance of the neural network system, we attempted to reduce this complexity by transforming the problem space.

The classification task was broken down into two successive parts, a top level classifier which identifies each input n-tuple of letters as one of four phoneme classes — vowels, diphthongs, consonants and the null phoneme [16, 28]. Appendix B.2 describes the classification of phonemes into such groups. This stage was followed by a separate neural network classifier trained specifically for each of the three groups containing more than one phoneme (viz. vowels, diphthongs and consonants). The selection of the classifier of the lower level was based on the outcome of the top-level classifier. The low-level classifier identified the data as belonging to an

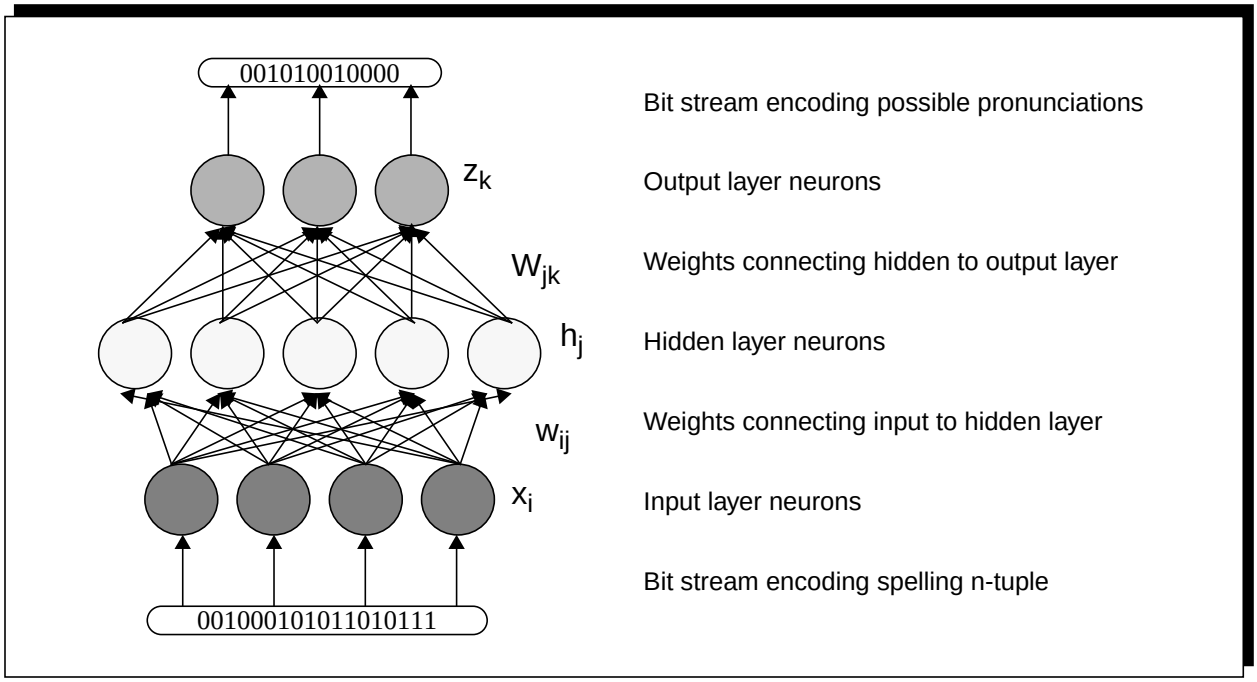

Figure 6. Illustration of a simple one-layered multilayered perceptron

individual phoneme belonging to the relevant sub-cluster of phones. As the number of phones under each group was reduced almost by a factor of 3, compared to the baseline system the complexity level at each classifier stage was reduced accordingly.

A block schematic for this system is displayed in Figure 5. This system was implemented with only one hidden neuron layer for each network as using multiple hidden layers did not affect performance significantly.

3.3. Non-stochastic Multilayered Perceptrons

To investigate whether there was a problem associated with the stochastic generation of pronunciations which caused the performance of the baseline system to degrade, we implemented a non-stochastic version of the neural network — a simple feedforward network. Since the multilayered perceptron (MLP) is an important constituent of the baseline system, we provide some theoretical fundamentals of this neural network.

The typical network has an input layer, an output layer, and at least one hidden layer. There is no theoretical limit on the number of hidden layers but typically there is just one or two. Figure 6 illustrates a typical such MLP with only one hidden layer. The input layer neurons are hardwired to the input bits x_i. These are connected through weights w_{ij} to the hidden layer of neurons, whose outputs are denoted by h_j. The hidden layer is connected to the output layer through the weights W_{jk}. The output of the output layers is the bit stream z_k while the desired output bits are y_k. For a network connected in this fashion, the j^{th} neuron in the hidden layer has an output

$$h_j = g\left(\sum_i w_{ij} x_i\right) \tag{1}$$

where *g(•)* is the activation function for the neuron. The outputs of the hidden layer are the inputs to the output layer neurons. For the k^{th} output neuron, the output is given by

$$z_k = g\left(\sum_j W_{jk} h_j\right) = g\left(\sum_j W_{jk} g\left(\sum_i w_{ij} x_i\right)\right) \tag{2}$$

Now we define an error or cost function for the network in terms of its parameters (weights) as

$$E[\boldsymbol{w}] = \sum_k (y_k - z_k)^2 = \sum_k \left(y_k - g\left(\sum_j W_{jk} g\left(\sum_i w_{ij} x_i\right)\right)\right)^2 \tag{3}$$

This is clearly a continuous differentiable function of all the weights in the network, and we can use a gradient descent algorithm to learn the appropriate weights that minimize this error function. Let η be the learning rate (a constant that determines the increment step for the weights). Then for the hidden layer - output layer connecting weights, the gradient descent yields an update term (also called the delta weight term) as

$$\Delta W_{jk} = -\eta \frac{\partial}{\partial W_{jk}} E[\boldsymbol{w}] = \eta \hat{\delta}_k h_j \qquad \textit{where} \qquad \hat{\delta}_k = (y_k - z_k) g'\left(\sum_j W_{jk} h_j\right) \tag{4}$$

and for the input layer - hidden layer weights the update terms are obtained as

$$\Delta w_{ij} = -\eta \frac{\partial}{\partial w_{ij}} E[\boldsymbol{w}] = \eta \delta_j x_i \qquad \textit{where} \qquad \delta_j = g'\left(\sum_j W_{jk} h_j\right) \sum_k W_{jk} \hat{\delta}_k \tag{5}$$

where *g'(•)* is the derivative of the activation function. This updating procedure is referred to as the backpropagation algorithm and is popularly used for training multilayered perceptron classifiers. Our stochastic system for generating multiple proper noun pronunciations borrows heavily from this concept.

The backpropagation learning can be modified by using different error functions, activation functions, and the modifying method of the derivative of the activation function. The weight updates can be performed with some ‘momentum’ to speed up learning, for instance, so that a fraction of the previous delta weight is fed through to the current update term. This essentially acts as a low-pass filter on the weight terms by reinforcing general trends but discouraging oscillatory behavior. Also, a cumulative weight update can be performed by accumulating the error for a number of input-output training pairs rather than updating every node for every input.

A major shortcoming of the backpropagation networks is that the internal representation of knowledge is not at all obvious or well-understood. Also, it is not guaranteed to converge and

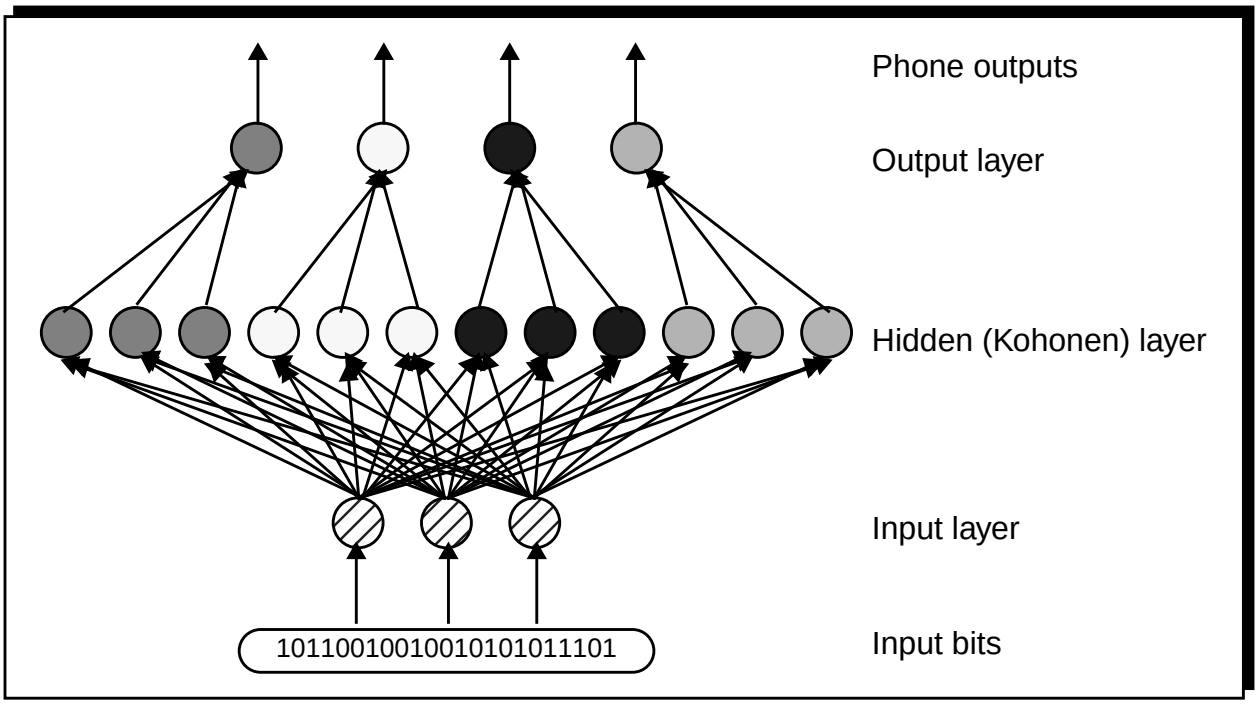

Figure 7. Topology of the learning vector quantizer — the groups of Kohonen neurons corresponding to different output classes are shown with identical shading.

suffers from the local minima problem. Also, backpropagation is a supervised algorithm and requires a large amount of training data.

3.4. Learning Vector Quantizer

A learning vector quantizer (LVQ) [32] is a single layer ANN which typically follows unsupervised learning. The network automatically adjusts its weights so that input patterns similar in some sense produce similar outputs. Thus the input patterns can be classified on the basis of the outputs they produce. The output neurons compete with each other to produce an output that best characterizes the input pattern. The LVQ is an ANN model used both for classification and segmentation problems. An example network is shown in Figure 7.

Topologically, the network contains an input layer, a single hidden layer called the Kohonen layer and an output layer. The output layer has as many neurons as the classes, while the Kohonen layer has a number of processing elements grouped for each of these classes. The number of processing elements per class depends upon the complexity of the input-output relationship. Usually, each class has the same number of elements throughout the layer. The Kohonen layer learns and performs relational classifications by deriving knowledge from the training set. Training is supervised, but the learning rules vary significantly from those used in backpropagation.

During training, neurons in the hidden layer are activated such that the distance of the training vector to each neuron (in terms of a scalar product of the weights with the input vector) is computed and the neuron with the smallest distance is declared the winner. The winner enables

only one output neuron to fire at a time, corresponding to the class the input vector belongs to. If the winning element is in the expected class of the training vector, its weights are reinforced towards the training vector. Otherwise the connection weights entering that neuron are moved away from the training vector. Thus, during the training process individual neurons in the hidden layer assigned to a particular class migrate to the region associated with their specific class. During the generation mode, the distance of an input vector to each hidden neuron is computed and again the nearest element is declared the winner. That in turn generates one output, signifying a particular class found by the network.

For complex classification problems such as pronunciation generation, the network requires a large Kohonen layer with many neurons per class. The training algorithm can also be enhanced by adding a conscience mechanism, boundary adjustments or an attraction function at different points while training the network [33].

4. STATISTICAL DECISION TREES

Tree structured classifiers are constructed by repeated splits of data into its descendent subsets based on some division criteria. The tree grows with each split until some stopping conditions are met or no split is possible. The terminal subsets form a partition of the original data set. Each terminal subset corresponds to a class (possibly more than one terminal subset may have the same class label). In a binary decision tree, each splitting criterion consists of a binary-valued condition (or a yes-no type of question). Typically, these questions are designed so as to maximize the entropy of the data given the division into various subsets. The size of the tree is determined by the total number of levels created by the splits. In some cases, pruning is used to get the right sized tree, i.e. one that has the lowest misclassification rate.

4.1. Fundamentals of Decision Trees

A decision tree is built from a set of learning samples, which consists of objects that are completely described by a set of attributes and a class label. The attributes can be ordered or unordered — ordered values are typically quantifiable and follow a natural ordering; while unordered values are typically logical and do not follow any particular order. For example, the age of a person is an ordered value while the gender is unordered. The tree can be termed as univariate or multivariate, depending on the number of attributes used as features at each internal node.

The typical topology of a decision tree is given in Figure 9. A decision tree contains a root node, zero or more internal nodes, and one or more leaf (or terminal) nodes. All internal nodes have one or more child nodes. All non-terminal nodes contain splits, which test the value of a mathematical or logical expression of the attributes. Edges from an internal node to its children are labelled with distinct outcomes of the test at that node. Each leaf node contains a class label, and the root node contains all the class labels. The number of classes is finite. A leaf node is said to be pure if all the training samples at that node belong to the same class.

The accuracy of the tree is determined by its misclassification rate i.e. the ratio of the samples misclassified to the total number of samples classified. The true misclassification rate of a decision tree is estimated from the training set or a held-out test set. If the training set is used to

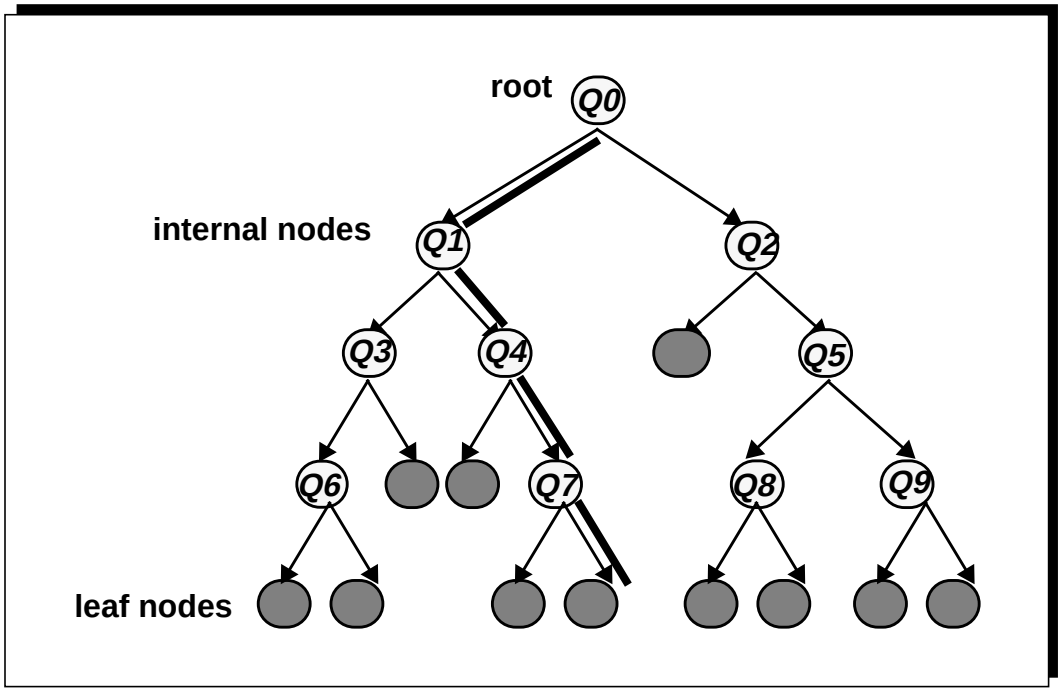

Figure 8. Typical topology of a statistical binary decision tree — the splitting starts at the root node and continues at each internal node according to the corresponding question. The terminal (or leaf) nodes indicate the classes.

estimate the true misclassification rate then it typically provides a quite optimistic estimate of the classification accuracy of the tree. This is often referred to as a *resubstitution estimate*. A more realistic estimate can be obtained by using some held-out test set, and it is often called a *test sample estimate*. Another metric of performance is the *k*-fold cross-validation of the tree. Here, the training data is divided into *k* equal-sized parts. Then each partition is held out one by one as the test set and the remainder is used to construct the tree. The misclassification rate is found in each case and finally averaged over all *k*.

4.2. Methods of Tree Construction

The manner of growing a tree differs from discipline to discipline, but the underlying structure is the same. There are roughly four heuristic categories of tree construction methods, viz. bottom-up, top-down, hybrid, and growing-pruning. These are briefly described in the following.

Bottom-up approach

In the bottom-up approach, a tree is constructed starting with the leaf nodes which are recursively merged to form the internal nodes and the root. The distances between classes are computed using some distance measure. Two classes with the smallest distance are merged to form a new group and create a new node. This process is repeated until only one group is left i.e. the root node is reached. In a tree constructed this way, the more obvious discriminations are done first closer to the root, while the more subtle ones are performed at later stages of the tree.

Top-down approach

For the top-down approach, the tree is constructed starting from the root node and splitting recursively until some stopping criterion is met. This process requires a set of binary questions, a goodness-of-split criterion that can be evaluated for a split at any node, a rule to stop splitting, and a rule to assign every terminal node to a class to construct the tree [34]. The set of questions generates a set of splits at every node. The splits are compared using some goodness measure, and the split that maximizes this goodness measure is selected.

Hybrid approach

The hybrid approach is a combination of the top-down and bottom-up approaches. The bottom-up approach is used to help the top-down approach in constructing the tree.

Growing-pruning approach

In the growing-pruning approach, the tree is grown according to a top-down approach. However, there is no stopping rule, and the tree is grown to its maximum size. At this point, pruning is used to trim the tree to get to the right size.

4.3. Splitting Rules

A significant part of the research in statistical decision trees is directed towards methodologies for creating an optimal set of attributes that best discriminate the input data; and in finding appropriate decision rules that use these attributes and split the data into corresponding classes. Since a decision tree is an estimator that is driven by the training data, it uses heuristic feature evaluation rules that aim to produce as reliable an estimate from the training data as possible. The decision rules can be roughly divided into three categories — rules derived from information theory, those derived from distance measures, and rules derived from dependence measures [35]. Not all rules used in decision trees fall into such distinct categories, often they are derived using a mixture of these metrics.

Information theoretic rules

Rules derived from information theory are variations based on the maximum entropy concept. One such rule maximizes the mutual information — the tree is constructed such that each split contributes to the largest gain in the average mutual information of the entire tree [36, 37]. Let the average mutual information obtained about a set of classes C_k from the observation of an event X_k, at a node k in a tree T be defined as

$$I_k(C_k;X_k) = \sum_{C_k}\sum_{X_k} p(C_{ki};X_{kj})\log\left[\frac{p(C_{ki}/X_{kj})}{p(C_{ki})}\right] \tag{6}$$

Event X_k represents the measurement value of a feature selected at node k and has two possible

outcomes; these measurement values are compared with a threshold associated with that feature at that node. Then the average mutual information between the entire set of classes C and the partitioning tree T can be expressed as

$$I(C;T) = \sum_{k=l}^{L} p_k I_k(C_k;X_k) \tag{7}$$

where p_k is the probability of the class set C_k and L is the number of internal nodes in the tree T. The probability of misclassification, p_e, of a decision tree classification T and the average mutual information $I(C;T)$ are also related as

$$I(C;T) \leq -\sum_{j=1}^{m} p(C_j)\log p(C_j) + p_e \log p_e + (1-p_e)\log(1-p_e) + p_e \log(m-1) \tag{8}$$

with equality corresponding to the minimum required average mutual information for a prespecified probability of error. Then a goal for design of the tree could be to maximize the average mutual information gain at each node k. The algorithm terminates, when the tree average mutual information, $I(C;T)$, exceeds the required minimum tree average mutual information specified by the desired probability of error.

Another rule maximizes the per node information gain, where each split contributes maximally to the reduction of the entropy at that node [36, 38]. Even though information gain provides good results, it has a deficiency of a strong bias in favor of tests with many outcomes. In other words, when one of the attributes contains unique information for all of the data, partitioning any set of training cases on the values of this attribute will lead to a large number of subsets, each containing just one case. Since all of these one-case subsets necessarily contain cases of a single class, the information gain will be maximal, but yet quite useless [39].

Distance measure rules

A diverse set of distance measures can be used to derive the tree-splitting rules. For instance, the Gini index of diversity emphasizes equal sized subgroups on splitting and attempts to maximize statistical purity of the children nodes[41]. Bhattacharya distance, Kolmogorov-Smirnoff distance, and χ^2 statistic are some of the classical distance measures that can be also used in deriving rules for tree construction.

A **twoing rule** divides a conglomeration of classes into two superclasses so that the problem can be considered as a two-class problem, where the greatest decrease in node impurity is realized. Twoing attempts to group together a large numbers of classes based on some common attribute near the top of the tree, and isolates individual classes near the bottom of the tree. It is an intuitive criterion that attempts to inform the user of class similarities [34]. The twoing criterion for any

node t and split s into a left node, t_L, and right node, t_R is defined by

$$\Phi(s,t) = \frac{p_L p_R}{4}\left[\sum_j \left|p(j|t_L) - p(j|t_R)\right|\right]^2 \tag{9}$$

The split that maximizes the twoing criterion at a node is determined as the best split for this node. For a discrete attribute, twoing investigates each possible combination of values resulting in two superclasses. For continuous attributes, the data is sorted and the midpoint between each data sample is used as the sample split. Once the twoing criterion is maximized, the split defined by this function is applied to the node to create two subsets of the data.

Bayesian splitting

Bayesian splitting uses a recursive partitioning algorithm to divide the training sample space into subsets based on some attribute [40]. A set of possible tests is applied to the current node, and the posterior probability contributed by the new leaves created using the test split is calculated. The test that yields the maximum posterior probability W is chosen for the actual splitting.

For C being the number of classes, V the number of partitions created by the test, ϕ_{jk} the relative frequency of occurrence

$$\phi_{jk} = n_{jk} \Big/ \sum_{j=1}^{C} n_{jk} \tag{10}$$

and $n_{jk} = \dfrac{number\ of\ class\ j\ elements\ in\ partition\ k}{number\ of\ total\ elements\ in\ partition\ k}$, the posterior probability is calculated by

$$W_{max} = max\ p_{jk}(W_V) = max \sum_{k=1}^{V}\sum_{j=1}^{C} n_{jk}\log\phi_{jk}. \tag{11}$$

4.4. Pruning Methods

A fully grown tree does not necessarily provide the best discrimination among the various classes of data. Often, it is useful to grow a complete tree and then remove subtrees that do not contribute significantly towards the generalization accuracy of the tree. This procedure is referred to as pruning the tree; and it is arguably better than the greedy splitting-stopping criteria since it can partially compensate for the suboptimality of greedy tree induction [34]. There are various strategies available for pruning depending on the nature of the problem, and selection of an appropriate technique usually requires some amount of heuristics.

A commonly used pruning technique is **cost-complexity pruning** where a sequence of increasingly smaller trees is built from the training data and the tree that has the highest classification accuracy on some held-out pruning test set is chosen as the pruned tree.

Let t_L and t_r be any two terminal nodes in the unpruned tree descending from the same parent node. The cost of a terminal node belonging to class j is defined as

$$R(t) = \frac{1}{n}\sum_{n} x_n \qquad \ldots x_n \notin j \tag{12}$$

If the cost of the parent node is equal to the sum of the cost of each child, i.e. $R(t_{parent}) = R(t_L) + R(t_R)$, then we can prune this node to get a new tree T_1 which is a subtree of the original tree.

The cost of a subtree is defined as

$$R(T_t) = \sum_{t \in T} R(t) \tag{13}$$

and its complexity is given by

$$\left|\tilde{T}\right| = \sum_{t \in T} t \qquad \ldots t \textit{ is a terminal node} \tag{14}$$

Next, we determine the node in T_1 that minimizes the cost-complexity parameter α given by

$$\alpha = \frac{R(t) - R(T_1)}{\left|\tilde{T}_t\right| - 1} \tag{15}$$

The node that minimizes α is the "weakest link" of subtree T_1 and should be pruned next, with T_2 being the resulting tree. Continuing in this same manner until no further pruning is possible, we can generate a decreasing sequence of subtrees in the order $T_1 > T_2 > \ldots > T_n$.

Another popular pruning strategy is called reduced-error pruning, in which the error rates of the tree and its components are assessed directly on a pruning set and the pruning that results in minimum error is selected [44].

4.5. Decision Trees for Pronunciations

In our initial attempts at applying a statistical decision tree to the problem of generating pronunciations for proper nouns, we constructed a univariate binary decision tree using the top-down approach. At each node, a series of questions was used to generate numerous splits of the data, and the split with the maximum entropy was chosen for that node. There was no stopping rule used, rather the tree was allowed to grow until it reached a maximum length. A basic snapshot of such a tree is shown in Figure 9.

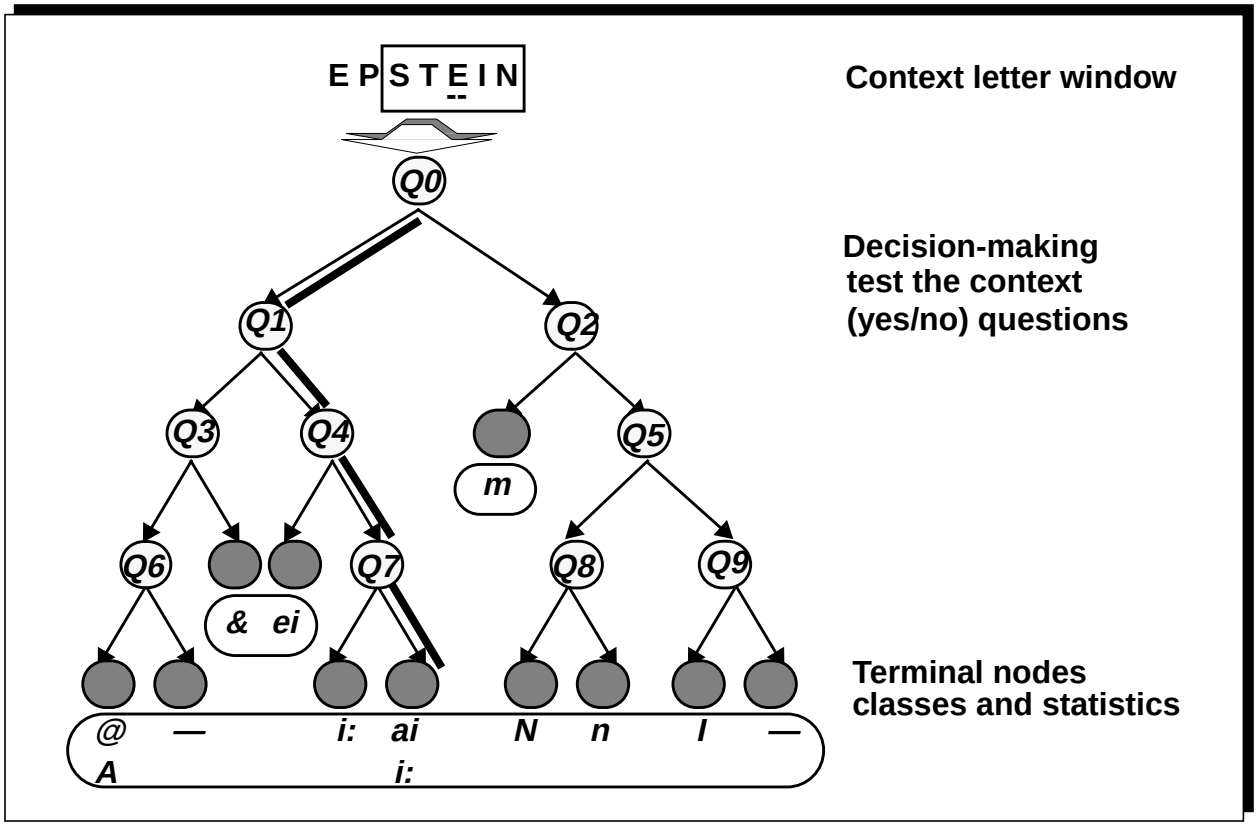

Figure 9. A typical statistical decision tree for automatic generation of pronunciations of proper nouns

The input to the decision tree was a name-pronunciation dictionary, where each spelling-pronunciation entry was converted into an n-tuple of letters centered at successive letters of the name and the corresponding phoneme, as described in Figure 10. The letter n-tuples were created using a sliding window of a fixed context length, and they constituted the attributes for the classes represented by the phonemes. The questions at each node are designed to exploit the contextual information embedded in the attribute window of letters. If $\{a_j\}$ is the set of all possible orthographic characters that can constitute the spelling and X_i is the i^{th} position in the context window of length n, then at each node the questions are of the form

$$X_i^? = a_j \qquad \ldots\ i = 0, 1, \ldots, n-1 \tag{16}$$

Thus there are a possible n questions to be asked at each node. The data is split according to the question X_i that has the maximum entropy according to the twoing rule. Let $C = \{1, 2, \ldots, J\}$ be the union of all the classes that the data can be grouped into. Then at each node, the data is separated into two subclasses $C_1 = \{1, 2, \ldots, J\}$ and $C_2 = C - C_1$ in the twoing strategy of splitting. For any given split s of the node n, the reduction in the node impurity $\Delta i(s, n)$ is

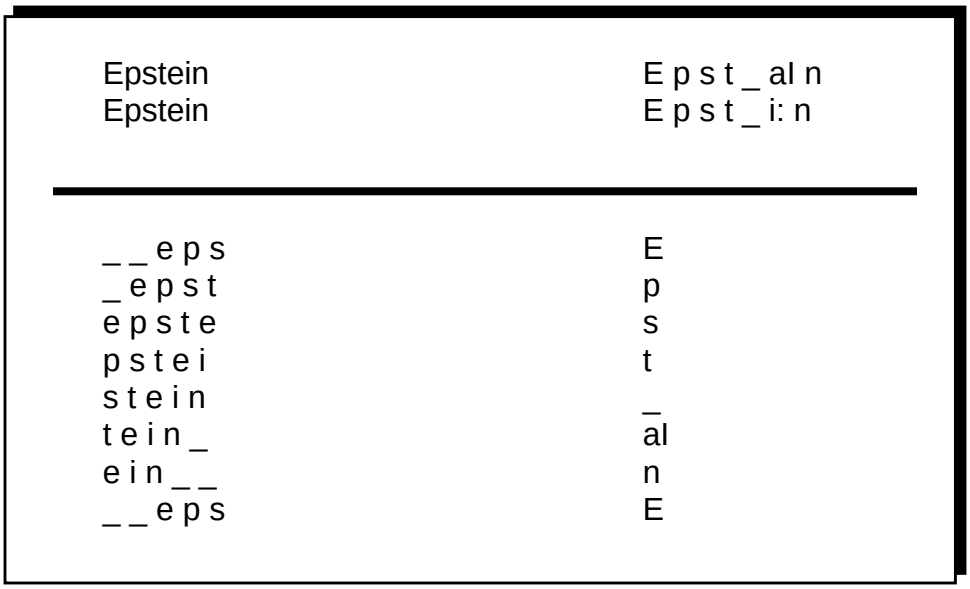

Figure 10. Example of the data used to train the decision tree — the name-pronunciation entries from the pronunciation dictionary (top) are converted to 5-letter context attributes with the phoneme corresponding to the middle letter as the class label.

computed for this two-class situation. For each such split, the node impurity is a direct function of the corresponding clustering performed on the data. The split *s* which maximizes this reduction in node impurity is used to partition the data. This twoing process is then repeated for both classes for further splitting, and so on till each class is pure.

5. UPDATES ON THE USER INTERFACE

The various modifications made in the pronunciation system were also reflected in an updated graphical user interface (GUI) built on top of the GUI used for the baseline system. The new Tcl-Tk based interface preserves the features of its predecessor, such as allowing the user to clear the display area, enter nouns for pronunciation generation and execute the system to obtain a network of phonemes that constitute the N-best list of pronunciations.

In addition, the next-generation interface allows the user to select the system from the different topologies / algorithms implemented — the stochastic multilayered neural network described in section 3.1, the multilayered perceptron system described in section 3.3 and the LVQ system depicted in section 3.4. Another choice is to use the decision-tree system. The user also has a choice of different context lengths for the input letter n-tuples. Moreover, audio capability has been added to the GUI so that the user can listen to the generated pronunciations and gauge their relative accuracy. We have used *rsynth*, a public-domain text-to-speech system for synthesizing the pronunciations into audio. A screen snapshot of the modified GUI is displayed in Figure 11.

6. EVALUATION OF THE MODIFIED SYSTEM

The performance of the pronunciation system and the various algorithms implemented therein

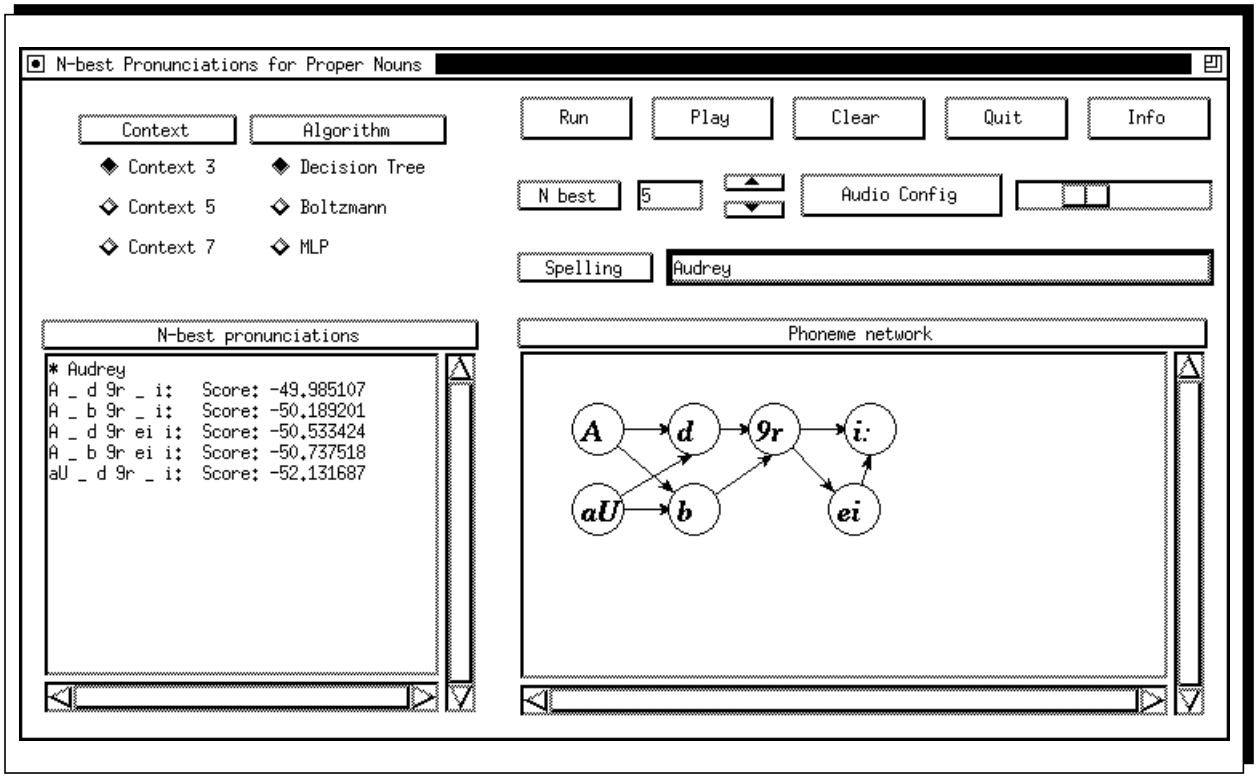

Figure 11. A screen snapshot of the graphical user interface for the updated system for generating multiple pronunciations for proper noun

was evaluated according to its success in generating accurate pronunciations of the input proper nouns. The system output was deemed to be erroneous if none of the pronunciations generated were found in the dictionary database from which the test name was derived. This case was dubbed as *no_correct*. The correct outcomes of the system were further subclassified under *all_correct* and *some_correct*, depending on respectively whether all the pronunciations generated matched with those in the dictionary or if some incorrect pronunciations were generated in addition to the correct ones. The scoring is performed automatically by a software that compares the hypothesis list of pronunciations to the corresponding proper noun entry in the dictionary.

Training and evaluation databases

Any statistical classification system relies heavily on a comprehensive training database to effectively model the discriminant features and characteristics of the problem domain. Likewise, a comprehensive dictionary of proper nouns along with their various pronunciations is an essential requirement for training the pronunciation generation system, both for the neural network as well as the decision tree application.

We have currently developed a database containing 18494 surnames found in the USA and corresponding 25648 pronunciations. This database adheres to the Worldbet pronunciation standards and represents a reasonably diverse set of names from a wide variety of ethnic origins. We have strived to achieve a reasonable mix of commonly occurring 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. The surnames have been collected from a variety of sources, notably [2, 45, 46]. The pronunciations are hand-transcribed in a modified subset of Worldbet phonetic transcription set. Automatic alignment is performed on each name-pronunciation pair using a dynamic programming technique and blank phoneme symbols '_' inserted in appropriate places to establish a one-one correspondence between letters and phonemes. Figure 10 provides an excerpt from the dictionary.

Design of experiments

To achieve a comprehensive benchmark of performance we divided the complete data set into a training set and a test set. The training set consists of 15000 names selected at random from the dictionary and their corresponding pronunciations. The remaining 3494 names constitute the test set. This division of data was done three times to create three different training sets (which have some overlap with each other) and three different test sets (which do not overlap with each other or the corresponding training set at all). Each training set consisted of approximately 20000 pronunciations corresponding to the 15000 names. The n-best pronunciation generating systems were trained and evaluated on each of the three data sets over various sets of system parameters, thus producing three benchmarks for each case. We found no significant difference in these over any parameter set and therefore will present only the overall results averaged over the three data sets.

Given the time requirements of training the neural network system on the full data set (a network with 1000 neurons single hidden layer takes approximately 8 to 12 hours per iteration on a 200 MHz Pentium processor), we devised a preliminary data subset for conducting evaluations in an efficient and timely manner. We selected a subset of the pronunciations dictionary consisting of all 2022 name-pronunciation pairs corresponding to the 1665 four-lettered surnames. The training set derived out of this comprised of 1331 surnames, while the rest were held out for evaluations. All evaluations were conducted for a 3-best list of pronunciations, unless specified otherwise.

6.1. Neural Network Systems

We followed a simple-to-complex approach in evaluating the various versions of the neural network system. Each enhancement in the system was first tested using simple experiments with synthetically generated alphabet strings, to provide us with a broad idea of the basic functioning of the system and its dependence on various parameters. The subsequent experiments were conducted on a subset of the real data (proper nouns and their pronunciations) to obtain an initial benchmark derived only from the 4 and 5 letter-long surnames. Based on the performance on these pilot tests the system was scaled up to handle the larger, complete database.

Multiple layered stochastic networks

A number of evaluation experiments were conducted on the stochastic multilayered network to determine the relationship of the change in topology and various modifications in the training algorithms. First a test was performed to study the effect of multiple layers using a small data set of synthetically created alphabet strings. The data for this pilot test consisted of two sets — one

containing 4-letter alphabet strings belonging to two classes (e.g. class 1 — aaaa, abaa, baaa etc. and class 2 — bbbb, bbab, bbba etc.), and the other having 5-letter strings similar to the previous case but corrupted with any letter (e.g. class 1 — aaaxa, ayaha, gdaaa etc. and bqbbs, ppbbb etc.). In each experiment, the network was trained with a start temperature of 100 and an exponential cooling rate of 0.1 per iteration. The results of this evaluation are depicted in Table 1.

The next step was to conduct similar benchmarks with real name-pronunciation data. The training schedule was kept the same as before for this task. The training algorithm was modified to use an adaptive learning rate for updating each weight. The pronunciation error rates for some of the significant experiments in this set are recorded in Table 2. As can be seen, there is no appreciable improvement in performance with multiple layers. Moreover, training the multilayered network took considerably longer per iteration compared to the single layered network. Therefore, we decided to run further evaluations only with the single hidden layer systems.

# hidden layers	# neurons per hidden layer	# training iterations	context length	pronunciation generation error rate %		
				all_correct	some_correct	no_correct
1	125	90	4	77.27	22.73	0
2	2 2			0	77.27	22.73
	16 2			0	86.36	13.64
	2 16			50.00	0	50.00
	16 4			0	54.55	45.45
	16 8			71.28	28.72	0
3	8 8 2			34.18	53.56	12.26
1	105	45	5	46.00	54.00	0
2	25 10			76.00	24.00	0
3	12 8 4			63.00	30.00	7.00

Table 1: Result of pilot experiments (two-class alphabet string classification) with multilayered stochastic neural networks. The no_correct performance indicates% error, the other two error rates (all_correct and some_correct) add up to the system accuracy.

eval data	# hidden layers	# neurons per hidden layer	# training iterations	no_correct pron generation %	
				training data	test data
4-letter names	1	1000	100	46.88	55.69
	2	128 64		45.29	52.13
	2	256 32		47.03	55.98
full training and test sets	1	1024		52.73	57.16
	1	2048	60	48.18	52.34
	2	256 32		49.32	55.27

Table 2: Evaluations of the multilayered stochastic network with context length of 5. The performance with multiple layers is not much different from the unilayer case.

It is significant to note that the performance of the network maintained its level when applied to the full data set, unlike prior cases where there was a complete breakdown in performance. We attribute this fact to the enhancements in the system that allow it to model the complex statistics of the data in a better fashion.

Bilevel hierarchical network

We conducted evaluations on this system with the training data set tagged according to the phoneme type — vowels, diphthongs, consonants and the null vowel (see Appendix B.2) for the high-level classification, and subsets of the data belonging to each of these four classes for the low-level classifier. Experiments were run both on the four-letter subset of the data as well as the

data set used	# neurons per hidden layer	# training iterations	context length	phoneme category classification error %	
				training data	test data
four letter surnames	100	150	7	11.27	22.46
	150			10.97	21.86
	200			12.02	22.75
	500			11.50	23.65
full	150			34.83	39.89

Table 3: Results for top-level classification for the hierarchical classifier network

classifier level	classifier type	# neurons in hidden layer	# train iterations	context length	classification error %	
					train data	test data
high	phoneme category	300	150	7	9.98	10.31
low	vowel	400			36.69	37.58
	diphthong	400			12.40	13.08
	consonant	800			63.94	64.11
overall generation of pronunciations for above system					71.86	73.18

Table 4: Results for the bilevel classifier neural network on the four-letter subset of the surname database

full data set (of 18494 surnames). As mentioned earlier, only single hidden layer networks were used. The results of the top-level classification are depicted in Table 3.

A complete evaluation was run on the four-letter names set with somewhat encouraging results (see Table 4). In this case, the training algorithm used was the one with adaptive learning constants. It was observed that the high-level classification and the low-level classification for diphthongs work quite; but the low-level classification performance in case of vowels and consonants is not very good. This highlights the confusability of vowels (by virtue of multiple and overlapping pronunciations depending on context) and consonants respectively.

# hidden layers	# neurons per hidden layer	# training iterations	context length	no_correct pron. generation %	
				training data	test data
1	300	500	5	28.25	66.91
	500	100		22.24	76.72
	1000	50		33.15	79.41
	2000			28.88	75.00
	4000			29.56	74.51
	8000			30.86	75.25
2	300 150			35.72	79.16
2	500 300			34.81	78.85

Table 5: Results of the evaluation tests on (non-stochastic) multilayered perceptrons using the 4-letter surnames database.

Multilayered perceptrons

Given the time constraints involved with training and evaluating a multilayered stochastic network on the real pronunciations data, we were motivated to evaluate a non-stochastic version of the system — an MLP — which would train faster. This system is essentially one-best i.e. it has only one output possible for any given input. The training algorithm was modified to run more efficiently, by encoding the activation of neurons in terms of various energy intervals and performing the weight updates on a word-to-word basis rather than a letter-by-letter basis. Also, the threshold term for each neuron was added separately as opposed to being included implicitly as part of the other weights. We conducted a number of evaluations with this system, the results of which are summarized in Table 5.

Learning vector quantizer

An elementary implementation of the LVQ was evaluated using the four-letter dataset with discouraging results as the network error function diverged during training. The situation is to be investigated further for more practicable implementations of the LVQ algorithm.

6.2. Decision Tree System

A similar series of experiments was run on the decision tree system. To test the applicability of the approach to classify clusters of letter strings, initial evaluations were run on alphabet string classification, on the data described earlier in Section 6.1. Encouraged by these results (as described in Table 6), we constructed a decision tree to implement the bilevel classifier described in Section 6.1. The performance of this system on the four-letter surnames data is summarized in Table 7. As can be seen, the decision tree system is excellent in memorizing the statistics of the input data. However, it lacks the ability to generalize this statistical knowledge to previously unseen situations, and therefore performs rather poorly on the test data subset.

Finally, a complete evaluation on the full data set was carried out using different length context windows. A similar phenomenon was observed in this case as well — the performance of the decision tree was close to perfect on the training data, but dropped by orders of magnitude when extended to the test data. Moreover, the performance degraded consistently with increasing context length. This is expected in light of the failure of the tree to generalize, as with a larger

context length	decision tree pruning strategy	% classification error	
		training data	test data
3	none	0.00	0.00
5		0.00	1.00
7		0.00	3.00

Table 6: Alphabet string classification results using an elementary decision tree classifier trained on 200 alphabet strings and tested on 100 strings.

classifier level	classifier type	pruning strategy	context length	classification error %	
				training data	test data
high	phoneme category	none	7	6.45	8.39
low	vowel			20.19	59.19
	diphthong			3.48	39.91
	consonant			2.58	49.18
overall generation of pronunciations for above system				16.30	63.92

Table 7: Performance of the bilevel classifier decision tree on the four-letter surnames database

context length the number of possible letter combinations increases exponentially and reflects in an increasing proportion of letter n-tuples not represented by the training set that can appear in the test set. The results of this evaluation for 5-best and 10-best generation of pronunciations are tabulated in Table 8.

6.3. A Comparison of the Two Approaches

As can be observed from the results described in Section 6.1 and Section 6.2, it can be concluded that while the neural networks generate accurate pronunciations better on small data sets, their performance fails to scale up as much when the network is applied to train and test on the full data set. However, the performance of the system is consistent on unseen data, due to the ability of the network to generalize. On the other hand, the decision tree system performs comparatively better on the full data; but the pronunciation generation accuracy suffers considerably when the decision tree system encounters previously unseen names.

The two approaches also contrast in their training requirements. On the same data set, decision

context length	pruning strategy	n-best pronunciations	pronunciation generation error %	
			training data	test data
3	none	5	34.78	47.13
5			10.23	66.29
7			8.17	87.25
3		10	31.26	42.53
5			4.31	61.37
7			2.83	83.11

Table 8: Performance of the decision tree system on the full data set

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 — block	NETtalk with block-decoding post-processor	78.00
NETtalk — legal	NETtalk — a connectionist network	67.00
Neural network	multilayered feedforward network	74.00
Decision tree	twoing criteria for node-splitting	82.00

Table 9: Comparative performance of various name-pronunciation systems on a 400 surname test set (courtesy [47]) compared to performance of our systems on a similar test set of 400 names.

trees train very fast and with no significant cost of computations. They also require very large memory resources for storing the node-wise splits and probabilities. Neural networks, on the other hand, need a large amount of computationally intensive training, but have very little by way of memory requirements.

A comparative performance evaluation of various rule-based systems is provided in [47]. Since we do not have access to the training and test data used for this benchmark, it is difficult for us to compare our system performance against this benchmark. However, by selecting test data from our dictionary in an analogous manner, we attempted to emulate the test conditions and conducted an evaluation (see Table 9). Even though a valid comparison is not possible, we estimate our system performance to be comparable to many of the rule-based systems. The decision-tree based system, especially has the potential to do a lot better with more sophisticated architecture and training (splitting, pruning and smoothing) algorithms.

7. CONCLUSIONS

We have implemented a number of modifications in the Boltzmann machine-based stochastic neural network system developed to automatically generate n-best pronunciations of proper nouns. We have also explored alternate approaches to tackle this problem, most notable among them being a statistical decision tree system. Both systems require only the text-based spelling of the proper noun as input to generate a list of likely pronunciations.

While the performance of the system has shown significant improvements compared to its predecessor, the pronunciation error rate on large test data sets still leaves a lot to be desired. Issues such as selection of the neural network parameters (number of hidden layers, number of neurons in each of them, training schedule — start temperature and cooling rate) are yet unresolved. Similarly, designing more appropriate splitting questions for the implementation of the decision tree, and application of a suitable pruning strategy to avoid overspecification of data are issues that demand further research. These are complex problems that depend on

application-specific quantities such as the amount of training data, the size of the context used and the stopping criteria used in training.

While the system implementation is a matter that deserves further research, the pronunciation dictionary is a valuable accomplishment of this project. The dictionary, currently consisting of 18494 surnames and 25648 pronunciations, is the only resource of its kind providing multiple possible pronunciations for various proper nouns. We also intend to augment this database with more names in the future. We expect the pronunciation dictionary to be a useful resource to the entire speech research community in this regard.

The publications generated during the course of this project, the pronunciation dictionary, as well as all the software developed in the course of this project is available in the public domain at *http://www.isip.msstate.edu/resources/technology/projects/1997/nbest_pronunciations/*.

8. FUTURE RESEARCH DIRECTIONS

Both the approaches discussed in this report, viz. neural networks and decision trees, have certain characteristics that display their potential for solving the problem of automatic pronunciation modeling of proper nouns. However, there are certain obvious shortcomings that need to be overcome to achieve this goal. We envisage our future research efforts to be directed towards the development of more powerful systems that employ these techniques.

We intend to apply more powerful training paradigms and examine newer topologies for the stochastic neural network system. An efficient implementation of a probabilistic learning vector quantizer with discriminant training will be such a topology to be implemented. The decision tree implementation will be enhanced with more discriminating questions and splitting strategies, as well as prudent usage of pruning techniques.

9. ACKNOWLEDGMENTS

We wish to thank Dr. Jack Godfrey at Texas Instruments for his numerous contributions in the development of the training database, and his continued support of this project. We also wish to acknowledge the continual support and guidance we have received from our sponsors, Texas Instruments, particularly, Dr. Raja Rajasekaran and Dr. Vishu Vishwanathan. We are grateful to Dr. Barb Wheatley of the National Security Agency for her invaluable guidance and support in development of the surname pronunciations database, and the in the design of the original system.

10. REFERENCES

[1] 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.

[2] B. Wheatley and J. Picone, “Integrating Speech Technologies For Medical Applications,” presented at the Medical Applications of Voice Response Technology Conference in Pittsburgh, PA, Dec. 1989.

[3] 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.

[4] A.J. Vitale, "An Algorithm for High Accuracy Name Pronunciation by Parametric Speech Synthesizer", in *Journal of Computational Linguistics*, Vol. 17(3), 1991.

[5] M.F. Spiegel and M.J.Macchi, "Synthesis of Names by a Demisyllable-Based Speech Synthesizer (Orator)", in *AVIOS Journal*, Vol. 7, Special RHC/RBOC issue, 1990.

[6] C.H. Coker, K.W. Church and M.Y. Lieberman, "Morphology and Rhyming: Two Powerful Alternatives to Letter-to-Sound Rules for Speech Synthesis", in *Conference on Speech Synthesis, European Speech Communication Association*, 1990.

[7] A.R. Golding and P.S. Rosenbloom, "The Evaluation of Anapron: A Case Study in Evaluating a Case-based System", in *Working Notes of the AAAI-94 Workshop on Case-Based Reasoning*, pages 84-90, Seattle, WA, 1994.

[8] H.M. Meng, S. Seneff, and V.W. Zue, "Phonological Parsing for Reversible Letter-to-Sound/Sound-to-Letter Generation," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, pp. II-1-II-4, Adelaide, Australia, April 1994.

[9] G. Hinton and J. Anderson (Eds.), *Parallel Models of Associative Memory*, Erlbaum Associates, NJ, 1981.

[10] 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.

[11] T.J. Sejnowski and C.R. Rosenberg, ""NETtalk: A Parallel Network That Learns To Read Aloud," Tech. Rep. JHU/EECS-86/01, John Hopkins University, Baltimore, MD, 1986.

[12] M.D. Riley, "A Statistical Model for Generating Pronunciation Networks", *in Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing*, S11.1, pp. 737-740, Toronto, Canada, May 1991.

[13] J.L. Elman and D. Zipser, "Learning the Hidden Structure of Speech", ICS Report 8701, University of California at San Diego, 1987.

[14] A. Waibel, H. Sawai and K. Shikano, "Modularity and Scaling in Large Phonemic Time-delay Neural Networks", *in IEEE Transactions on Acoustics, Speech and Signal Processing*, Vol. 30, pp. 1888-1898, December 1989.

[15] B.J. Wheatley and J. Picone, "Voice Recognition of proper nouns Using Text-Derived Recognition Models," US Patent No. 5212730, May 18, 1993.

[16] N. Deshmukh, M. Weber and J. Picone, "Automated Generation of N-Best Pronunciations of Proper Nouns", in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, Atlanta, GA, May 1996.

[17] F. Rosenblatt, "The Perceptron: A Perceiving and Recognizing Automaton", Cornell Aeronautical Laboratory Report 85-460-1, 1957.

[18] M.L. Minsky and S. Papert, *Perceptrons*, MIT Press, Cambridge, MA, 1969.

[19] J.J. Hopfield, "Neural Networks and Physical Systems with Emergent Collective Computational Abilities", in *Proceedings of the National Academy of Sciences USA*, Vol. 79, pp. 2554-2558, 1982.

[20] R. Rosenfeld, "A Hybrid Approach to Adaptive Statistical Language Modeling," in *Proceedings ARPA Workshop on Human Language Technology*, pp. 76-81, Plainsboro, NJ, March 1994.

[21] T. Robinson, M. Hochberg, and S. Renals, "IPA: Improved Phone Modeling With Recurrent Neural Networks," in *Proceedings IEEE International Conference on Acoustics, Speech, and Signal Processing*, pp. I-37-40, Adelaide, South Australia, Australia, April 1994.

[22] A. Waibel, T. Hanazawa, G. Hinton, K. Shikano, and K. Lang, "Phoneme Recognition Using Time-Delay Neural Networks," in *IEEE Transactions on Acoustics, Speech, and Signal Processing*, Vol. 37, no. 3, pp. 328-339, March 1989.

[23] N. Weir, U.M. Fayyad, and S. Djorgovski, "Automated Star/Galaxy Classification for Digitized POSS-II," in *The Astronomical Journal*, Vol. 109, pp. 2401, 1995.

[24] P.E. File, P.I. Dugard, and A.S. Houston, "Evaluation of the Use of Induction in the Development of a Medical Expert System", in *Computers and Biomedical Research*, Vol. 27, pp. 383-395, October 1994.

[25] S. Shimozono, A. Shinohara, T. Shinohara, S. Miyano, S. Kuhara, and S. Arikawa. "Knowledge Acquisition from Amino Acid Sequences by Machine Learning System BONSAI", in *Transactions of the Information Processing Society of Japan*, Vol. 35, pp. 2009-2018, October 1994.

[26] L. Spirkovska, "Three Dimensional Object Recognition Using Similar Triangles and Decision Trees" in *Pattern Recognition*, Vol. 26, pp. 727, May 1993.

[27] D. Bowser-Chao and D.L. Dzialo, "Comparison of the Use of Binary Decision Trees and Neural Networks in Top Quark Detection", in *Physical Review D: Particles and Fields*, Vol. 47, pp. 1900, March 1993.

[28] N. Deshmukh and J. Picone, "Automatic Generation of N-best Proper Noun Pronunciations", ISIP Technical Report, Institute for Signal & Information Processing,

Mississippi State University, 1996.

[29] J. Hertz, A. Krogh and R.G. Palmer, *Introduction to the Theory of Neural Computation*, Addison-Wesley, Redwood City CA, 1993.

[30] D.E. Rumelhart, G.E. Hinton and R.J. Williams, "Learning Internal Representation by Error Propagation", in D.E. Rumelhart and J.L. McClelland (Eds.), *Parallel Distributed Processing — Vol. 1: Foundations*, MIT Press, Cambridge, MA, 1986.

[31] S. Kirkpatrick, C.D. Gellatt and M.P. Vecchi, "Optimization by Simulated Annealing", in *Science*, Vol. 220, pp. 671-680, 1983.

[32] T. Kohonen, "Automatic Formation of Topological Maps in a Self-organizing System", in E. Oja and O. Simula (Eds.), in *Proceedings of the 2nd Scandinavian Conference on Image Analysis*, pp. 214-220, 1981.

[33] T. Kohonen, "Learning Vector Quantization", in *The Handbook of Brain Theory and Neural Networks*, pp. 537-540, The MIT Press, Cambridge, MA, 1995.

[34] L. Breiman, J.H. Friedman, R.A. Olshen, C. Stones, *Classification and Regression Trees*, Wadsworth, Monterey, CA, 1984.

[35] M. Ben-Bassat, "Use of Distance Measures, Information Measures, and Error Bounds on Feature Evaluation", in P.R. Krishnaiah and L.N. Kanal (Eds.), *Classification, Pattern Recognition and Reduction of Dimensionality*, Vol. 2 of *Handbook of Statistics*, North-Holland Publishing Company, Amsterdam, 1987.

[36] S.R. Safavian and D. Landgrebe, "A Survey of Decision Tree Classifier Methodology", in *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. 21, No. 3, pp. 660-674, May 1991.

[37] I. K. Sethi and G. Sarvarayudu, "Hierarchical Classifier Design Using Mutual Information," *IEEE Transactions on Pattern. Analysis and Machine Intell*igence, Vol. PAMI-4, pp. 441-445, 1982.

[38] J. L. Talmon, "A Multiclass Nonparametric Partitioning Algorithm," in E. S. Gelsema and L. N. Kanal (Eds.), in *Pattern Recognition in Practice II*, Elsevier Science, Amsterdam, Netherlands, 1986.

[39] J. R. Quinlan, *C4.5: Programs for Machine Learning*, Morgan Kaufmann Publishers, 1993.

[40] W. Buntine, *A Theory of Learning Classification Rules*, Ph.D. thesis, University of Technology, Sydney, Australia, 1991.

[41] R.J. Light and B.H. Margolin, "An Analysis of Variance for Categorical Data", in *Journal of American Statistical Association*, Vol. 66, pp. 534-544, 1971.

[42] S. K. V. Murthy, *On Growing Better Decision Trees from Data*, Ph.D. thesis, Johns Hopkins University, Baltimore, MD, 1997.

[43] C. Shaffer, "Conservation of Generalization: A Case Study", Technical Report, Department of Computer Science, CUNY/Hunter College, February 1995.

[44] J.R. Quinlan, "Probabilistic Decision Trees", in R.S. Michalski and Y. Kodratoff (Eds.), *Machine Learning: An Artificial Intelligence Approach — Volume 3*, Morgan Kaufmann, San Mateo, CA, 1990.

[45] E.C. Smith, *American Surnames*, Genealogical Publishing Co. Inc., Baltimore, MD, 1986.

[46] R.A. Cole, M. Fanty and K. Roginski, "A Telephone Speech Database of Spelled and Spoken Names," in *Proceedings of the International Conference on Spoken Language Processing*, Banff, Alberta, Oct. 12-16, pp. 891-893, (1992).

[47] A.R. Golding and P.S. Rosenbloom, "A Comparison of Anapron with Seven Other Name-Pronunciation Systems", in *Journal of the American Voice Input/Output Society,* Vol. 14, pp. 1-21, August 1993.

APPENDIX A. TRAINING THE NEURAL NETWORK SYSTEM

We describe here the backpropagation-with-simulated-annealing algorithm used for training the stochastic neural network system with multiple hidden layers described in Section . We present the derivation for the simple case of one set of weights connecting two layers. The extension to multiple sets of connecting weights (more layers) is trivial.

Beneath the simulated annealing and error backpropagation the primary weight-updating algorithm is essentially a gradient descent method, where 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.

A.1. Derivation of The Weight-Update Rules

Let $\{w_{ij}\}$ be the set of weights connecting units in the two layers, such that $\{h_i\}$ 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

$$\Psi = \sum_{\hat{\alpha}} P_{\hat{\alpha}} \log \frac{P_{\hat{\alpha}}}{P_\alpha} \tag{A.1}$$

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

$$E_\alpha = \sum_j \sum_i w_{ij} h_i^\alpha h_j^\alpha \tag{A.2}$$

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

$$\frac{\partial e^{-E_\alpha / T}}{\partial w_{ij}} = \frac{1}{T} e^{-E_\alpha / T} h_i^\alpha h_j^\alpha \tag{A.3}$$

The probability that the system under thermal equilibrium conditions ends up in the global state α

is given by the following equation —

$$P_\alpha = \frac{e^{-E_\alpha/T}}{\sum_\lambda e^{-E_\lambda/T}} \tag{A.4}$$

where λ is any global state of the network. Differentiating Equation (A.4) and substituting Equation (A.3) we get

$$\frac{\partial P_\alpha}{\partial w_{ij}} = \frac{\frac{e^{-E_\alpha/T}}{T}}{\sum_\lambda e^{-E_\lambda/T}} \left(h_i^\alpha h_j^\alpha - \frac{\sum_\lambda h_i^\lambda h_j^\lambda e^{-E_\lambda/T}}{\sum_\lambda e^{-E_\lambda/T}} \right) \tag{A.5}$$

which simplifies to

$$\frac{\partial P_\alpha}{\partial w_{ij}} = \frac{P_\alpha}{T} \left(h_i^\alpha h_j^\alpha - \frac{\sum_\lambda h_i^\lambda h_j^\lambda e^{-E_\lambda/T}}{\sum_\lambda e^{-E_\lambda/T}} \right) \tag{A.6}$$

We use this relation to compute the gradient of the error function. Taking derivative of Equation (A.1) we get

$$\frac{\partial \Psi}{\partial w_{ij}} = -\sum_{\widehat{\alpha}} \frac{P_{\widehat{\alpha}}}{P_\alpha} \frac{\partial P_\alpha}{\partial w_{ij}} \tag{A.7}$$

Using Equation (A.6) we get

$$\frac{\partial \Psi}{\partial w_{ij}} = -\frac{1}{T} \sum_{\widehat{\alpha}} \frac{P_{\widehat{\alpha}}}{P_\alpha} P_\alpha \left(h_i^\alpha h_j^\alpha - \frac{\sum_\lambda h_i^\lambda h_j^\lambda e^{-E_\lambda/T}}{\sum_\lambda e^{-E_\lambda/T}} \right) \tag{A.8}$$

Noting the facts 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 —

$$\sum_{\widehat{\alpha}} P_{\widehat{\alpha}} = 1 \qquad ; \qquad h_i^{\alpha} = h_i^{\widehat{\alpha}} \tag{A.9}$$

If we term the error in the output bit as $\delta_j^{\alpha} = h_j^{\widehat{\alpha}} - h_j^{\alpha}$, then on further simplification of Equation (A.8) we get the relationship between the gradient of the error function and the bit error.

$$\frac{\partial \Psi}{\partial w_{ij}} = -\frac{1}{T}\left(h_j^{\alpha} - h_j^{\widehat{\alpha}}\right)h_i^{\alpha} = \frac{1}{T}\left(h_j^{\widehat{\alpha}} - h_j^{\alpha}\right)h_i^{\alpha} = \frac{1}{T}\delta_j^{\alpha} h_i^{\alpha} \tag{A.10}$$

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.

$$\Delta w_{ij} = \eta \delta_j^{\alpha} h_i^{\alpha} \tag{A.11}$$

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 we can minimize the noise in incrementing the weights, 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

$$\Delta w_{ij} = \eta \delta_j^{\alpha} h_i^{\alpha} + \mu \Delta w_{ij} \tag{A.12}$$

which is used in training the pronunciation generating system.

A.2. 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 multiple layers that maps the inputs onto the corresponding outputs.

Algorithm:

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 $k-1^{th}$ 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 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.
2. For $t = 0$, initialize the weights to random values between -0.1 and 0.1. Set the initial values of the momentum $\alpha(0) = 0$ and the temperature $T(0) = T_0$. The initial temperature T_0 is a parameter specified by the user.

For $t = 0, 1, 2, \ldots$

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.

$$\Delta E_{1_j} = -\sum_{i=0}^{N_0} w_{1_{ij}} x_i \qquad ; \qquad p(h_{1_j} = 1) = \frac{1}{1 + e^{\Delta E_{1_j}/T(t)}} \tag{A.13}$$

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 $k = 2,3,\ldots,(K-1)$ —

$$\Delta E_{k_j} = -\sum_{i=0}^{N_{k-1}} w_{k_{ij}} h_{k-1_i} \qquad ; \qquad p(h_{k_j} = 1) = \frac{1}{1 + e^{\Delta E_{k_j}/T(t)}} \tag{A.14}$$

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

$$\Delta E_{K_j} = -\sum_{i=0}^{N_{K-1}} w_{K_{ij}} h_{K-1_i} \quad ; \quad p(o_j = 1) = \frac{1}{1 + e^{\Delta E_{K_j}/T(t)}} \tag{A.15}$$

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}.

$$\delta_{K_j} = o_j(1 - o_j)(y_j - o_j) \tag{A.16}$$

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 $k+1^{th}$ layer and is denoted by δ_{k_j}. For all $k = K-1,\ldots,2,1$ —

$$\delta_{k_j} = h_{k_j}(1 - h_{k_j}) \sum_{i=0}^{N_k} \delta_{k+1_i} w_{k+1_{ij}} \tag{A.17}$$

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 $k = K, K-1,\ldots,2,1$ —

$$\begin{aligned} \Delta w_{k_{ij}} &= \eta \delta_{k_j} h_{k-1_j} + \alpha(t) \Delta w_{k_{ij}} \\ \Delta w_{1_{ij}} &= \eta \delta_{1_j} x_j + \alpha(t) \Delta w_{1_{ij}} \end{aligned} \tag{A.18}$$

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 most common cooling schedule for such networks follows an exponential function. The cooling exponent β is specified by the user to customize the training schedule.

$$\alpha(t) = 1 - e^{-\beta t} \quad ; \quad T(t) = T_0 e^{-\beta t} \tag{A.19}$$

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.

APPENDIX B. ALPHABET AND PHONEME SETS

The pronunciations generation system is designed to accept a total of 30 characters as input. These consist of the 26 letters of the English alphabet, as well as some special characters such as white space, apostrophe, hyphen and period. Both uppercase and lowercase alphabet symbols are accepted and treated in an identical fashion i.e. map to the same bit-strings in the input buffers.

The output of the pronunciations generating system for each input set is one of 46 different phonemes. The phonemes are transcribed in the Worldbet symbol set. Each phoneme is encoded by a bit-string that corresponds to the output bits of the outermost layer of the network.

The following tables summarize the symbols along with their bit-string equivalents. A typical word example is also given with the phoneme set to illustrate the pronunciation it represents.

B.1. Input Alphabet Set

Character symbol	Bit-string
	00000
-	00001
’	00010
.	00011
a	00100
b	00101
c	00110
d	00111
e	01000
f	01001
g	01010
h	01011
i	01100
j	01101
k	01110

Character symbol	Bit-string
l	01111
m	10000
n	10001
o	10010
p	10011
q	10100
r	10101
s	10110
t	10111
u	11000
v	11001
w	11010
x	11011
y	11100
z	11101

B.2. Output Phoneme Set

The letters that constitute the phoneme are indicated in italics.

Class	Phoneme	Example
Vowels	i:	B*ee*k
	I	St*i*ll
	E	G*e*tty
	@	P*a*stor
	A	H*aw*k
	^	L*u*nd
	>	M*o*ll
	&	Jud*a*
	U	M*oo*n
	u	S*u*mo
	oU	Cl*o*se
	ei	K*a*ne
Diphthongs	aI	B*y*ron
	>i	P*oi*rot
	aU	P*ow*der
	iU	L*iu*
	j	*Y*eats
	w	*W*alker
	l	Mu*l*der
	9r	T*r*avis
	3r	B*ur*ton
	>r	P*or*ter
	&r	Bank*er*

Class	Phoneme	Example
Consonants	b	*B*owles
	p	To*p*kins
	t	Mor*t*on
	d	Har*d*y
	k	*K*ellogg
	g	Rea*g*an
	h	*H*olmes
	v	La*v*er
	D	Fea*th*er
	T	*Th*omas
	s	Jame*s*on
	z	Ro*s*e
	S	Fi*sh*er
	Z	O*jh*a
	f	Ra*f*ter
	m	La*m*bert
	n	Parto*n*
	N	Ri*ng*er
	dZ	*J*ackson
	tS	Ar*ch*er
	ks	Ma*x*well
	&k	M*c*Donald
Null	_	