An Approach to Automatic Phonetic Baseform Generation Based on Bayesian Networks

Changxue Ma and Mark A. Randolph

Human Interface Laboratory
Motorola Labs
1301 E. Algonquin Rd. Schaumburg, IL, 60196
{Changxue.Ma, Mark.Randolph}@motorola.com

Abstract

To improve the performance and the usability of the speech recognition devices, It is necessary for most applications to allow users to enter new words or personalize words to the system vocabulary. Voice-tagging technique is a simple example that use speaker dependent spoken sample to generate baseform transcriptions of the spoken words. More sophisticated techniques can use both spoken samples and texts of the new words to generate baseform transcriptions. In this paper, we propose a new approach to the problem. We use Bayesian networks to model the letter-to-sound rule probabilities. Compared to the common decision tree based method, This new approach shows a definite advantage.

1. Introduction

Substantial progress has been made in voice-enabled systems. One of the key enablers are speech recognizers that can be configured dynamically by the application. An essential element of this configuration is the recognizer's pronunciation dictionary. In the past, dictionaries would be created and/or tuned by phoneticians. In emerging applications for over-the-telephone network-based recognition using VoiceXML and in hand-held devices, there is a significant need for pronunciation rules that are highly accurate when tuning by hand is impractical.

In this paper, we describe a probabilistic approach to phonemic baseform generation and introduce the use of Bayesian Networks for computing the key probability relations. We compare our new method with Decision Trees and conclude this paper with experimental results.

2. Phonetic Baseform Generation

Similar to other approaches, we address the problem of generating phonetic baseforms in two stages. The first stage requires solving the letter-to-sound prediction problem for each element in the input word or letter string. The second stage is to cascade the phoneme predictions resulting from the letter-to-sound stage using Dynamic Programming.

2.1. Letter-to-sound Prediction

Letter-to-sound prediction is a classical problem. Linguists have sought to codify letter-to-sound rules for a language using a system of *context-dependent rewrite* rules of the form

$$\alpha \rightarrow \beta / \gamma_1 \_ \gamma_2 . \quad (1)$$

In Rule (1), the symbol α denotes a letter taken from the input word string. The symbol β represents a phoneme and is α's corresponding *phonemic realization*. The symbols γ_1 and γ_2 are strings over the input symbol alphabet consisting of zero or more letters.

In early text-to-speech synthesis systems, pronunciation rules of the form represented in (1) were generated by hand. In a TTS system, the rules were part of the system's pronunciation module and were used in producing pronunciations for words not covered in the dictionary. Because the conversion of letters to sounds in languages like English is non-deterministic, for speech recognition systems and modern systems for text-to-speech synthesis, data driven methods using decision trees or neural networks have proven more effective.

2.2. Pronunciation Generation for Speech Recognition

For speech recognition, it is very helpful to frame the pronunciation problem in probabilistic terms. For a given orthographic word or letter sequence, $w = \alpha_1 \alpha_2 \cdots \alpha_N$, we'd like to obtain a corresponding pronunciation $s = \beta_1 \beta_2 \cdots \beta_M$, that maximizes the joint probability

$$P(s, w, O) = P(O, s \mid w) P(w) = P(O \mid w, s) P(s \mid w) P(w),$$

where O represents a sequence of acoustic observations, and $P(O \mid s, w)$ is the likelihood of this acoustic observation given the word w and its corresponding pronunciation s.

During phonetic recognition, we calculate the maximum likelihood phonetic sequence by solving

$$\hat{s} = \arg\max_s P(O \mid s, w) P(s \mid w). \quad (2)$$

Equation (2) is solved in two stages. In the first stage, we compute a pronunciation, s, for each word, w, in the vocabulary along with its corresponding probability, $P(s \mid w)$. Once the pronunciation dictionary is generated, Viterbi search is used to determine the phonetic sequence, $\hat{s}$, that optimizes $P(O \mid s, w)$.

2.2.1. DP-based Pronunciation Generation

This paper focuses on the problem of generating phonemic baseforms from letter sequences or the pronunciation problem. We are using a technique based on dynamic programming.

The terms s and w are strings over the phoneme set and alphabet respectively; the probability $P(s \mid w)$ can be rewritten in expanded form as

$$P(s \mid w) = P(\beta_1, \beta_2, \ldots \beta_n \mid \alpha_1, \alpha_2, \ldots \alpha_m) \approx \prod_i P(\beta_i \mid \alpha_1, \alpha_2, \ldots \alpha_m)\, P(\beta_i \mid \beta_{i-1}, \ldots \beta_1). \quad (3)$$

Expectation of Equation (3) clearly reveals that the principal challenge in letter to sound prediction is managing the complexity of the pronunciation model. Along the line of complexity reduction, the approximation made in (3) uses the Markov assumption of conditional independence. In particular, to compute $P(s \mid w)$ we use n-grams of the form $P(\beta_i \mid \beta_{i-1}, \ldots \beta_1)$, to model constraints on phoneme sequences, so called phonotactic constraints. Specifically, in our experiments, we used a bi-gram phonotactic model, $P(\beta_t \mid \beta_{t-1})$.

2.2.2. The importance of Context

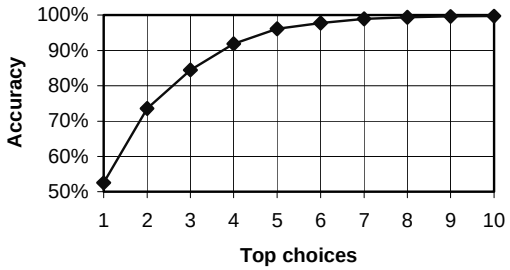
Low dimensional phonotactic models are of great assistance in managing the complexity of computing $P(s \mid w)$. However, the complexity of this computation is dominated by the term

$$P(\beta_i \mid \alpha_1, \alpha_2, \ldots \alpha_m),$$

which simply states that the sound predicted for a particular letter, is a function of the letter and a suitable amount of its context.

This context dependence is illustrated in Figure 1, where we have plotted the prediction accuracy experimentally obtained by using only the focus letter in the letter-to-sound conditional probability calculation, i.e., by only using the expression $P(\beta_t \mid \alpha_t)$ in evaluating Equation (3).

Figure 1: Phoneme prediction accuracy from letter-to-sound rules without use of context.



2.3. Decision Trees

The letter-to-sound conditional probability can be re-written as

$$P(\beta_i \mid \alpha_{t-K} \ldots \alpha_{i-1}, \alpha_i, \alpha_{i+1} \ldots \alpha_{i+K}). \quad (4)$$

where the conditioning variables have been rearranged in a manner to draw attention to the *focus letter* and its *left* and *right* context. Clearly, for long context strings (where K is large), estimating (4) becomes intractable. Like others, the approach that we take is to limit K to be less than 2, and beyond that, apply some additional measure to reduce the number of parameters that remain.

In the past, investigators have used decision trees to cluster contexts into statistically meaningful equivalence classes. During training, a learning sample is partitioned into sub-samples using a set of questions about the properties of the focus letter's context, such as, *"Is the letter at a word end?"* or *"Is the letter followed by a consonant?"*

The skillful use of Decision Trees in this application is largely a matter of i*)* framing the right set of questions and *ii)* growing the right-sized tree. Determining the proper set of questions can be viewed as an art, and typically requires linguistic knowledge of the language and lots of experiments. In growing the right-sized tree, there is an inherent trade-off between the classification performance on training data and the performance obtained on input letter strings outside the training set.

The leaf nodes of the decision tree represent the above-mentioned equivalence classes for letter-to-sound prediction. The letter-to-sound conditional probability is readily obtained as the size of the majority class in a leaf node divided by the node population. Because this probability is determined on the basis of counts, the tree-size estimation problem is directly related to the robustness of these probability estimates. Trees are typically grown to a size too large. A pruning strategy is employed to merge nodes upward in the tree to produce smoothed probability estimates. Probability estimates that are over-smoothed will result in unsatisfactory performance in the actual system. In experimental results presented below, this behavior is demonstrated.

3. Bayesian Networks

In this paper we introduce Bayesian Networks as an alternative to Decision Trees for reducing the complexity of the letter-to-sound conditional probability. Rather than reducing the number of contexts by clustering them equivalence classes, Bayesian Networks exploit certain conditional independence and decompose $P(\beta_i \mid \alpha_{t-K} \ldots \alpha_{i-1}, \alpha_i, \alpha_{i+1} \ldots \alpha_{i+K})$. into simpler terms that can be estimated using less data.

3.1. Theory

Bayesian Networks exploit the property of conditional independence: if random variables X and Y are independent given Z, then

$$P(X \mid X, Y) = P(X \mid Z). \quad (5)$$

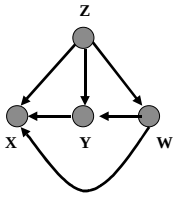
For probability expressions having very large number of variables, if appropriate conditional independence assumptions hold, there can be a substantial reduction in the number of probabilities that are needed.

3.1.1. Probability Relations represented as Directed Acyclic Graphs

Bayesian Networks provide a language for representing independence statements based on directed acyclic graphs (or DAG's). Consider a more complicated probability relation,

$$P(X, Y, W, Z) = P(X \mid Y, W, Z)P(Y \mid W, Z)P(W \mid Z)P(Z). \quad (6)$$

Figure 2: Bayesian Network corresponding to P(X,Y,W,Z).



The DAG corresponding to this expression is shown in Figure 2. In comparing the DAG in Figure 2 to the right hand side of (6), one note that nodes and arcs in the graph represent the conditional probabilities or terms on the right hand side of the above expression. In general, in a Bayesian network, nodes correspond to random variables. A node, n_i, will have predecessor nodes $\{n_i, n_{i-1}, \ldots n_{i-k}\}$ to represent the statistical dependence between the arguments off $P(n_i \mid n_{i-1}, n_{i-2}, \ldots n_{i-k})$. Conversely, conditional independence statements are represented by graphs where arcs have been pruned. This is illustrated in Figure 3.

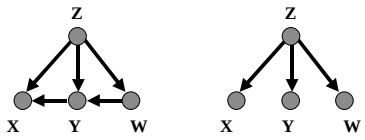
Figure 3(a) shows the case where random variable X is independent of random variable W given the conditioning variable Z, corresponding to the following equation.

$$P(X, Y, W, Z) = P(X \mid Y, Z)P(Y \mid W, Z)P(W \mid Z)P(Z).$$

The degenerate case of conditional independence in which all random variables are conditionally independent is the so-called Naïve Bayesian Network shown in Figure 3(b). This latter figure represents the equation

$$P(X, Y, W, Z) = P(X \mid Z)P(Y \mid Z)P(W \mid Z)P(Z).$$

Figure 3: Examples of Bayesian Networks illustrating conditional independence.



3.2. Bayesian Networks applied to Letter-to-sound prediction

Our application of the theory of Bayesian Networks to the letter-to-sound prediction problem is enabled by the assumption that context dependencies tend to be *local*. Put in more practical terms, the pronunciation of letters at the beginning of a word is largely independent of the letter-to-sound relations at the word's end. Furthermore, for reasonably long words, if there is dependence, then it is most likely accidental and not easily modeled by any statistical method. Thus, the kinds of probabilistic independence assumptions that allow us to reduce the number of parameters represented by Bayesian networks will probably apply to the letter-to-sound prediction problem.

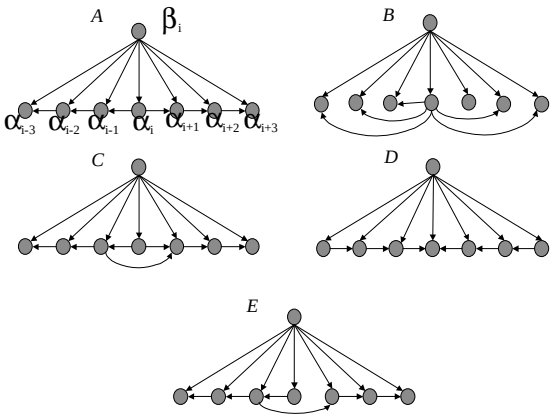
Figure 4 shows a number of network topologies examined in our experiments. Note that in all cases, we have limited the context to three letters on each side of the focus letter (i.e., K=3). Furthermore, none of the graphs are fully connected. In each, conditional assumptions have been made.

3.3. Learning Bayesian Networks

Learning Bayesian networks from data is a very active research area. Its goal is to use heuristic search techniques to derive a network topology over all possible network topologies which best describes the probability distributions of the training data.

In there investigation reported in this paper, we did not focus on the network learning problem. In particular, for networks shown in Figure 4, A through D were determined by handed. Network E was determined automatically via a learning mechanism proposed by Chow and Liu [6] and extended by Friedman [7]. This algorithm starts with a Naïve Bayes network and incrementally adds arc until reasonable classification performance is obtained. Our work on network learning is preliminary; more results will be reported in a later paper.

Figure 4: Bayesian Networks Topologies for letter to sound rule.



4. Experimental results

In our experiments, CMU dictionary cmu58k94 containing 58,000 words was used as training data. The context width seven is used as shown in Figure 3. The total number of seven-letter context segment we have collected is 404405. We randomly selected a number of 322209 segments for training and the remained for testing purpose. It is worth noting that the training set and test set are mutual exclusive. This is very important to keep them mutual exclusive because non-parametric methods like decision tree method can have perfect performance for the test samples which contained in the training set.

The alignment is done incrementally. We started with a basic set of word being aligned manually or automatically. We sift through the whole training dictionary and select the most frequently wrong-predicted word segment. We correct them and add them to the training dictionary. We iterate the process until we reached a satisfactory result.

Table 1: *Phone prediction accuracy comparison for Bayesian Networks.*

Top choices		First	Second	Third
A	test	83.3	93.4	96.9
	train	85.6	94.9	97.8
C	test	84.7	94.9	97.6
	train	89.0	97.1	99.0
Naïve		69.94	85.49	91.69
B		78.3	91.2	96.02
D		84.3	94.3	97.0
E		80.01	89.91	92.81

In Table 1, we see better performance for the Bayesian network with higher order statistics. The middle node of the networks is closely tied to the neighboring nodes, such as networks *C* and *D*. The performances of network *B* and *E* are the lowest because the weak dependence of the context. Further more, It is noted that the performance decision tree, shown in Table 2, can be improved. However, this is based on sacrifice the performance on training data.

Table 2: *Phone prediction accuracy from decision tree method with tree pruning and without tree pruning*.

Top choices		First	Second	Third
Prune	test	82.58	87.58	88.58
	train	95.94	99.14	99.64
Not pruned	test	80.5	83.9	84.4
	train	97.80	99.9	100.0

5. Conclusions

In this paper, we propose to use Bayesian networks to model the letter-to-sound rule probabilities for automatic phonetic baseform generation. Compared to the common decision tree based method, This new approach shows its advantage over the decision tree based method and gives better performance.

6. References

[1] Bahl, L.R.; Das, S.; deSouza, P.V.; Epstein, M.; Mercer, R.L.;Merialdo, B.; Nahamoo, D.; Picheny, M.A.; Powell, J., "Automatic Phonetic Baseform Determination", *ICASSP-91*, p. 173 –176, vol.1, 1991.

[2] Haeb-Umbach, R. Beyerlein, P. and Thelen, E. "Automatic Transcription of Unknown Words In A Speech Recognition System", *ICASSP-95*, p. 840-843

[3] Weintraub, M. Wegmann, S. Kao, Y. H. Khudanpur, S. Galles, C. Fosler, E. Saraclar, M. "Automatic Learning of Word Pronunciation From Data", JHU Workshop, 1996.

[4] Meng, H.M.; Seneff, S.; Zue, V.W, "Phonological parsing for reversible letter-to-sound/sound-to-letter generation", *ICASSP-94*., Volume: ii , 1994 , p II/1 -II/4 vol.2.

[5] Pearl, J. *Probabilistic Reasoning in Intelligent Systems.* Morgan Kaufmann, 1988.

[6] Chow, C.K., and Liu, C.N. "Approximating Discrete Probability Distributions With Dependence Tree", *IEEE Transaction on Info. Theory* 14:462-467, 1968.

[7] Friedman, N., Geiger D., and Goldszmidt M. "Bayesian Network Classifier", *Machine Learning*, 29:131-163, 1997.