
AUTOMATIC GENERATION OF DETAILED PRONUNCIATION LEXICONS

Michael D. Riley and Andrej Ljolje

AT&T Bell Laboratories, Murray Hill, NJ 07974, USA

ABSTRACT

We explore different ways of “spelling” a word in a speech recognizer’s lexicon and how to obtain those spellings. In particular, we compare using as the source of sub-words units for which we build acoustic models (1) a coarse phonemic representation, (2) a single, fine phonetic realization, and (3) multiple phonetic realizations with associated likelihoods. We describe how we obtain these different pronunciations from text-to-speech systems and from procedures that build decision trees trained on phonetically-labeled corpora. We evaluate these methods applied to speech recognition with the DARPA Resource Management (RM) and the North American Business News (NAB) tasks. For the RM task (with perplexity 60 grammar), we obtain 93.4% word accuracy using phonemic pronunciations, 94.1% using a single phonetic pronunciation per word, and 96.3% using multiple phonetic pronunciations per word with associated likelihoods. For the NAB task (with 60K vocabulary and 34M 1-5 grams), we obtain 87.3% word accuracy with phonemic pronunciations and 90.0% using multiple phonetic pronunciations.

1 INTRODUCTION

Current practice in speech recognition is to form word models from the concatenation of sub-word units. In other words, each word is spelled in terms of some finite alphabet of acoustic units. These units are variously referred to as “phones”, “phonemes”, or “phoneme-like units”. How the spelling of a particular word is obtained is not often discussed. In particular, whether that spelling is a coarse, abstract phonemic representation or a finer, more concrete

phonetic one must be decided. In this article, we explore this issue and present recognition results comparing different solutions to this problem.

The phonemes are a set of baseforms for representing the sounds in a word. Replacing one phoneme in a word with another is usually drastic enough to turn that word into a different word (or a non-word). It is, by definition, drastic enough to do so for some word. There is finer kind of variation, however, that is not indicated at the phonemic level, so-called *allophonic variation*. Which allophone for a given phoneme will occur depends on factors like phonemic context, speaking rate, and the dialect of the speaker. Thus, phones are acoustically distinct realizations of phonemes. The choice of a phone set is a matter of judgment, but phoneticians have traditionally agreed that certain kinds of variation are worth noting based on their acoustic prominence and regularity.

A specific example will help clarify the issue. Consider the word **bottle**. Its *phonemic* spelling is /**b aa t ax l**/ (we use here ARPABET as the phonemic symbol set in examples [18]). For an American speaker, the /**t**/ in **bottle** will most likely be flapped [**dx**] when this word is uttered and the /**ax l**/ will most likely reduce to a syllabic [**el**]. So the most likely *phonetic* spelling of **bottle** is [**b aa dx el**]. In this paper, we use the TIMITBET symbols, a superset of the ARPABET symbols, for specifying phones [6].

Which spelling should we use for recognition purposes? If we use the phonemic spelling, we will, in effect, require the acoustic model, say, of /**t**/ to handle all its allophones, which include [**t**], [**dx**], and [**q**], and handle its deletion. On the other hand, if we use the phonetic spelling, we have the problem that if the speaker utters a likely, but not the *most* likely pronunciation of a word, e.g., **bottle** as [**b aa t el**], then there is a mismatch between the permitted and true pronunciations for that rendition.

There is, of course, another choice and that is to allow for multiple phonetic spellings. So [**b aa dx el**], [**b aa t el**], [**b aa dx ax l**], and [**b aa t ax l**] could all be allowed spellings of that word. This has the obvious cost of increasing the vocabulary size. It also raises the question of how many alternative spellings per word to allow. Too many and we risk recognizing an unlikely pronunciation of the wrong word over the likely pronunciation of the correct word; too few and we have a problem similar to using only the most likely pronunciation.

As a final refinement, we could attach a likelihood to each phonetic realization of the word, so **bottle** is pronounced as [**b aa dx el**] with 75% probability,

as [b aa t eɪ] with 13%, [b aa dx ax ɪ] with 10% , and [b aa t ax ɪ] with 2%.

We still have to decide how many alternative pronunciations to allow, but probably now it is purely an issue of computational speed and space, since including unlikely pronunciations should not hurt recognition accuracy if we take their likelihood into account. In this case, we have to have a scheme for not only finding the alternative pronunciations, but also for assigning likelihoods to them. In the following sections, we will examine each of these possibilities, describing methods for generating pronunciations and evaluating their performance.

While much speech recognition work has assumed a single pronunciation per word, usually obtained from a dictionary or constructed on-site by hand, there have been numerous exceptions to this rule. For example, some researchers have hand-constructed phonological rules systems that are applied on the lexical baseforms (especially at cross-word boundaries) with good result [7, 10]. Others have explored the automatic derivation of pronunciation alternatives and their probabilities by corpus-based methods [5, 19, 13, 2, 17], several using decision-tree based methods similar to those described below.

2 PHONEMIC PRONUNCIATIONS

Our first approach to the problem of how to spell a word for the purposes of speech recognition is to use a simple phonemic representation as found in an on-line version of the Collins American English Dictionary. When a word is not found in the dictionary, morphological rules and, failing those, letter-to-sound rules are brought to bear. We have, in fact, very briefly described the pronunciation component of the Bell Labs Text-to-Speech system, which is what we use to obtain the phonemic spellings [3, 4]. There is usually only one allowed pronunciation per word. An exception would be, for instance, **data** (/d ey t ax/ and /d ae t ax/).

We see that the generation of baseform spellings is done with a conservative strategy. Only if the word is not found in a hand-compiled dictionary, do we utilize letter-to-sound rules. This is because the mapping from English orthographic spelling to its base pronunciation is quite complex, while published dictionaries that tabulate this mapping for many words are readily available.

We shall use two of the widely-available tasks for our evaluations. The first is the DARPA Resource Management (RM) task with a vocabulary of 1,000 words and the simple word-pair grammar (perplexity 60) as the language model; we test on the February 1989 test set. The recognizer used in these experiments is described in detail in [15, 12, 11]. The second task is the DARPA North American Business (NAB) task with a vocabulary of 60,000 words and an n -gram grammar with 34 million 1-to-5 grams; we test on the November 1994 test set. This recognition system is described in [16].

Suffice it to say here that for both tasks we used state-of-the art acoustic modeling, comprising Gaussian-mixture hidden Markov models.

For the experiments on the RM task a two-pass strategy was used. First, an N -best list of sentences is produced using limited context-dependency. We find, that with just $N = 10$, the correct sentence is present 97% of the time with our best system.

In the second pass, we rescore these alternative sentences one at a time. With this scheme we have all contextual information readily available. Thus, in the rescoring we use full context-dependent models including across word boundaries. We are also able, as described below, to modify the phonetic spelling with the cross-word context. Since the correct sentence is almost always present in the N -best output of the first pass for this task, we incur little added search error with this convenient scheme.

With such a phonemic lexicon, we are able to achieve 93.4% word accuracy on the DARPA RM Feb '89 test set.

We also used a two-pass strategy for the NAB task. In the first pass a word lattice is generated using acoustic and language models which are cheap enough to use and accurate enough so that the correct answer is almost always present in the word lattice. The upper bound on the search errors caused by the loss of the correct answer in the word lattices is 0.2%.

In the second pass the word lattice is rescored using the most accurate acoustic and language models available. The acoustic models are similar in structure to the models used in the experiments on the RM test set.

With a phonemic lexicon we achieve 87.3% word accuracy on the DARPA NAB Nov '94 test set.

The recognition performance on both of the tasks is remarkable given the relative abstractness of the representation. Apparently, the HMM acoustic models with Gaussian mixture distributions we use are able to model to some degree the natural allophonic variation. Nonetheless, we shall see that we can do better.

3 THE PHONEME-TO-PHONE MAPPING

As indicated in the Introduction, we can refine the baseform *phonemic* pronunciations by using more detailed *phonetic* pronunciations. But how shall we determine, in general, appropriate phonetic realizations from the phonemic input? We will take some care to answer this question in this section before returning to their effect on recognition performance in Sections 4 and 5.

To get a measure of the difficulty of this problem, consider the TIMIT phonetically-labeled database. This is a hand-labeled corpus of 6300 phonetically-rich utterances by 630 speakers created at TI and MIT [6], and includes both phonetic and orthographic transcriptions of the utterances. From the orthographic transcriptions, we have derived phonemic transcriptions and aligned them with the phonetic transcriptions. From this, we have estimated the conditional entropy of a phone given the matching phoneme to be about 1.5 bits.

Significantly, this estimate does not include any contextual information. Knowing what the neighboring phonemes are, what the stress environment is, and where the word boundaries are, will help considerably in predicting how a phoneme is realized as a phone. Below we describe a method that will reduce the uncertainty from 1.5 bits to about .8 bits. In another way of measuring performance, this method predicts the correct phone from phonemic context about 83% of the time and the correct phone lies in our top five guesses 99% of the time. In comparison, if we only use the matching phoneme and no contextual information, we are able to predict the correct phone only 69% of the time and we must look at the top ten guesses to find the correct phone 99% of the time.

There are different approaches toward predicting pronunciations. The traditional phonetician's approach has been to write rules that explicitly state the predicted behavior, e.g., 'a stop consonant is usually unexploded before another stop consonant' or 'a /t/ usually becomes a glottal stop before a nasal in the same word' [9].

PHONEME	PHONE1	PHONE2	CONTEXT
b	1.00 b		
aa	0.92 aa		
t	0.71 dx	0.12 t	
ax	0.78 el	0.12 ax	
l	0.95 l		(if ax→ax)
	0.98 -		(if ax→el)

Figure 1 Pronunciation network for `bottle`. The first column gives the phoneme to realize. Pairs of probabilities and phones follow. For example, the phoneme /t/ is predicted to realize as a [dx] with 71% probability and as a [t] with 12% probability. Realizations with less than 10% probability are pruned from this figure. Some realizations depend on the realization of the previous phoneme. For example, the phoneme /l/ will delete with 98% probability if the previous /ax/ was realized as [el] but will appear with 95% probability as [l] if the /ax/ was realized as [ax]

We instead take a corpus-based approach. Now that large, phonetically-labeled databases like TIMIT are available, it is possible to estimate the phoneme-to-phone mapping statistically [14, 2, 17]. This approach has several advantages. First, it readily permits assigning likelihoods to alternative pronunciations (which we shall use latter). Second, it permits the discovery of regularities perhaps overlooked by heuristic means. Finally, it allows predictors to be quickly retailored to new corpora – whether different tasks, different dialects, or even different languages.

Figure 1 shows an example of the phone realization alternatives for the word `bottle` derived by methods described below. The first column gives the phoneme to realize. Pairs of probabilities and phones follow. For example, the phoneme /t/ is predicted to realize as a flap, [dx], with a 71% probability and as a [t] with probability 12%. Phone realizations with less than 10% probability are pruned from this figure. In the following sections, we will describe how we obtained such networks and will return to this example.

3.1 Problem Formulation

To predict from phonemes to phones, we take a phoneme string as input and produce phonetic realizations as output along with their likelihoods.

Let us make this idea precise. Let $\mathbf{x} = x_1x_2\dots x_m$ be the string of phonemes of some sentence. So that we can mark both word boundaries and stress we augment the phoneme set to include /#/ as a word boundary marker and split each syllabic phoneme into an unstressed, a primary stressed, and a secondary stressed version. Further, let $\mathbf{y} = y_1y_2\dots y_n$ be the string of corresponding phones. We include the phone symbol [-] to indicate that a phoneme may delete.

The most general form of our predictor is $\hat{P}(\mathbf{y}|\mathbf{x})$, where \hat{P} estimates the probability that the phone sequence \mathbf{y} is the realization of the phoneme sequence \mathbf{x} .

This specifies the probability of an entire phone sequence \mathbf{y} . For convenience, we want to decompose this into one phone prediction at a time. Since

$$P(\mathbf{y}|\mathbf{x}) = p_n(y_n|\mathbf{x}y_1\dots y_{n-1})p_{n-1}(y_{n-1}|\mathbf{x}y_1\dots y_{n-2})\dots p_1(y_1|\mathbf{x}) \quad (1)$$

we can restate the problem as finding a suitable predictor, $\hat{p}_k(y_k|\mathbf{x}y_1\dots y_{k-1})$, that estimates the probability that y_k is the k th phone in the realization, given the phoneme sequence \mathbf{x} and the previous $k-1$ phones $y_1\dots y_{k-1}$.

Eq. 1 is more general than necessary since realistically the k th phone will depend only on a few neighboring phonemes and phones. Suppose that we can place the phoneme and phone strings into alignment. In fact, forming a good alignment between phonemes and phones is easy if deletions and insertions are permitted, using a phonetic feature distance measure and standard string alignment techniques [8]. Since we have augmented the phone set to include a deletion symbol, the only stumbling block to such an alignment would be if phones insert. For the moment, assume that they don't; we will come back to insertions later. Thus, under this assumption we can talk about the k th phoneme and its corresponding phone. We assume

$$p_k(y_k|\mathbf{x}y_1\dots y_{k-1}) = p(y_k|x_{k-r}\dots x_{k-1}x_kx_{k+1}\dots x_{k+r}y_1\dots y_{k-1}). \quad (2)$$

In other words, p_k is stationary and depends only on the $\pm r$ neighboring phonemes.

If we assume the k th phone does not depend on any of the previous phones, we have

$$\begin{aligned} p(y_k|x_{k-r}\dots x_{k-1}x_kx_{k+1}\dots x_{k+r}y_1\dots y_{k-1}) \\ = p(y_k|x_{k-r}\dots x_{k-1}x_kx_{k+1}\dots x_{k+r}). \end{aligned} \quad (3)$$

This is the assumption that phones are conditionally independent given the phonemic context. A less stringent assumption would be that the k th phone

only depends on the immediately prior phone, In this case, we must estimate

$$\begin{aligned} p(y_k | x_{k-r} \dots x_{k-1} x_k x_{k+1} \dots x_{k+r} y_1 \dots y_{k-1}) \\ = p(y_k | x_{k-r} \dots x_{k-1} x_k x_{k+1} \dots x_{k+r} y_{k-1}). \end{aligned} \quad (4)$$

This is the assumption that phones are conditionally 1st-order Markov given the phonemic context.

These last two models are the ones that we will explore – one that assumes an independence model of phones and the other that assumes a Markov model. We must also come back to the question of what to do when phones insert.

3.2 Classification Trees

We now discuss the question of how, in general, we will estimate the phoneme-to-phone mapping probabilities specified in Section 3.1. The simplest procedure would be to collect n-gram statistics on the training data. A bi-phonemic or possibly tri-phonemic context would be the largest possible with available training data if we want statistically reliable estimates.

We believe that a straight-forward n-gram statistics on the phonemes are probably not ideal for this problem since the contextual effects that we are trying to model often depend on a whole class of phonemes in a given position, e.g., whether the preceding phoneme is a vowel or not. A procedure that had all vowels in that position clustered into one class for that case would produce a more compact description, would be more easily estimated, and would allow a wider effective context to be examined.

Thus intuitively we would like a procedure that pools together contexts that behave similarly, but splits apart ones that differ. An attractive choice from this point of view is a statistically-generated decision tree with each branch labeled with some subset of phonemes for a particular position. The tree is generated by splitting nodes that statistical tests, based on available data, indicate improve prediction, but terminating nodes otherwise.

An excellent description of the theory and implementation of tree-based statistical models can be found in *Classification and Regression Trees* [1]. The interesting questions for generating a decision tree from data – how to decide which splits to take and when to label a node terminal and not expand it further – are discussed in these references along with the widely-adopted solutions.

Suffice it to say here the result is a binary decision tree whose branches are labeled with binary cuts on the continuous features and with binary partitions on the categorical features and whose terminal nodes are labeled with continuous predictions (*regression tree*) or categorical predictions (*classification tree*). By a continuous feature or prediction we mean a real-valued, linearly-ordered variable (e.g., the duration of a phone, or the number of phonemes in a word); by a categorical feature or prediction we mean an element of an unordered, finite set. (e.g., the phoneme set).

3.3 Baseline Prediction Model

In Section 3.1 we developed two models for predicting the probability that a particular phone is the realization of a phoneme. One used the phoneme context as the predictor input. The other used the same plus the previous phone as input. Here, we describe the implementation of the first model. We still exclude the treatment of insertions at this point. We will call this our *baseline model*. In Sections 3.4 and 3.5, we will describe refinements to this model.

In the exposition in Section 3.1, we combined word boundary and stress information into the phoneme set itself. When we actually input the features into the tree classification procedure we have found it more convenient to keep them separate.

We include $\pm r$ phonemes around the phoneme that is to be realized (typically, $r = 3$). This is irrespective of word boundaries. We pad with blank symbols at sentence start and end.

Since there 40 different phonemes (in the ARPABET), if we directly input each phoneme into the tree classification routine, 2^{40} possible splits would have to be considered per phoneme position at each node, since, by default, all possible binary partitions are considered. This is clearly intractable, so instead we encode each phoneme as a feature vector. A manageable choice is to encode each phoneme as a four element vector: (**consonant-manner**, **consonant-place**, **vowel-manner**, **vowel-place**). Each component can take one of about a dozen values and includes ‘n/a’ for ‘not applicable’. For example, /s/ is encoded as (**voiceless-fricative**, **palatal**, **n/a**, **n/a**) and /iy/ is encoded as (**n/a**, **n/a**, **y-diphthong**, **high-front**)

If the phoneme to be realized is syllabic, then we also input whether it has primary or secondary stress or is unstressed. We use stress as predicted by the

Bell Labs text-to-speech system; this is essentially lexical stress with function words de-accented. If the phoneme is not syllabic, we input both the stress of the first syllabic segment to the left and to the right if present within the same word (and use ‘n/a’s’ if not).

To encode word boundaries, we input the number of phonemes from the beginning and the end of the current word to the phoneme that is being realized.

We do not input the syllabification directly since we do not have that information readily available. But, because we typically use a wide phonemic context, the syllabification is often implicitly present. If we had the syllabification, however, we would include it since it might help in some cases. We note, nonetheless, that Randolph [17], who included the syllabification in a tree classifier of TIMIT stop allophones, achieved classification rates nearly identical to what we achieve on that data using the feature set describe here.

Our output set is simply a direct encoding of the phone set plus the symbol [-] if the phoneme deletes. Computation time grows only linearly with the number of *output* classes so this direct encoding presents no problem similar to the exponential growth found with size of the input feature classes.

We now describe the results of this baseline model on the TIMIT database. The phonetic transcription of 3024 sentences from the TIMIT ‘sx’ and ‘si’ sentences were aligned with their phonemic transcription as predicted by the Bell Labs text-to-speech system from their orthographic transcription. For each of the resulting 100702 phonemes, the phonemic context was encoded as described in Section 3.3. A classification tree was grown on this data and the tree size was chosen to minimize prediction error in a 5-fold cross-validation. The resulting tree had approximately 300 nodes.

This tree was then used to predict phonetic realizations of an independent 336 sentences from the TIMIT ‘sx’ and ‘si’ sentences. The result was 84.1% correct prediction and a conditional entropy of .77 bits.

3.4 Markov Model

To judge the 84.1% performance obtained by our baseline model, we have to look at the errors. They can be divided into two categories: those in which the prediction was, in fact, the most likely outcome, but the speaker of the test sentence used a less likely alternative pronunciation (e.g. he didn’t flap

the /t/ in ‘pretty’), and those cases in which the model is imperfect and an implausible pronunciation is predicted. The first kind of uncertainty is inherent to the problem, the second kind is something we should try to fix.

Using the baseline model, we find the major latter kind of error occurs near a deletion. For example, ‘are’, phonemically /aa r/, is often realized as [axr] in fluent speech. In the training data this is modeled as /aa/ → [axr] and /r/ → [-]. But, the baseline model may predict the pronunciations [aa -] and [axr r] for /aa r/, which are unlikely for most TIMIT speakers. Similar problems occur with /n/, /m/ and /l/ in contexts where they are likely to syllabify.

The problem is that in these cases the realization of the previous phoneme strongly influences the realization of the current phoneme. This suggests we should use the second model outlined in Section 3.1 – the Markov model. The idea is that we augment the feature set with the previous phone that was output. During training, we use the actual phone uttered. During testing, we use dynamic programming to maximize Eq. 1 (with Eq. 2 & Eq. 4) over all phones.

We encode the previous phone with a scheme similar to that for phonemes, but add a few extra categories to fully specify all the phones.

The result is an improvement to 85.5% correct predictions on the TIMIT test set described in Section 3.3. Significantly, the implausible predictions like those described above near a deletion are judged to have very low probability with this newer model.

3.5 Treatment of Insertions

There are two ways to deal with the insertion of phones. The first way is add a second model that predicts the phone insertions. Consider a phone sequence $z_0 y_1 z_1 y_2 z_2 \dots y_n z_n$ that is the realization of phoneme sequence $x_1 x_2 \dots x_n$. We view phone y_i as the realization of phoneme x_i and view phone z_i as an insertion between phoneme y_i and y_{i+1} . In a realistic example, there will be only an occasional insertion, so most of the z_i insertions will be marked as [-]’s. This scheme allows only one phone to insert after a phoneme; however, it is clear this can be generalized. In practice, contiguous insertions seldom occur. For example, for 100702 TIMIT phonemes, there were 13907 single insertions but only 352 multiple insertions.

The second way to deal with insertions is to augment the output set to include phone pairs. For example, the phoneme /æ/ can be realized as the pair of phones [q æ]. The insertion tree approach accounts for this by treating one of these phones as an insertion. Instead, we might add [q+æ] to our ‘phone’ set. The advantage of this approach is we can use the methods described in Section 3.3 and 3.4 without any need to predict insertions separately. The potential disadvantage is that we could conceivably need to square the size of our output phone set.

Fortunately, in practice, only a few phone pairs are found commonly. For example, a glottal stop inserts before a vowel (e.g., phrase initially). We used the 37 most common phone pairs to augment our phone set. This set accounts for 95% of the insertion tokens in the TIMIT database.

A classification tree grown using this output set and the model of Section 3.4 predicts the realization of a phoneme correctly 83.3% of the time and has a conditional entropy of .82 bits. Note that the classification rate is lower here than in the previous models since this model must also predict phone insertions.

3.6 Discussion of Results

In the pronunciation network in Figure 1, for the word ‘**bottle**’, we see that some non-trivial transformations have been captured. In particular, the flapping of the /t/ and the realization of /uh l/ as a syllabic l. Further, note some realizations depend on the realization of the previous phoneme – the phoneme /l/ will delete with 98% probability if the previous /ax/ was realized as [e1] but will appear with 95% probability as [1] if the /ax/ was realized as [ax].

In the pronunciation network in Figure 2, for the words ‘**had your**’, we see that the phoneme /d/ in ‘**had**’ is predicted to realize as [jh] with 51% probability and as [d] with 37% probability. This is an example where an alternative pronunciation is quite likely. Further, as another example of a realization depending on a prior realization, the phoneme /y/ in ‘**your**’ will delete with 73% probability if the previous /d/ was realized as [jh] but will appear with 90% probability as [y] if the /d/ was realized as [d].

We can use dynamic programming to find efficiently the highest probability path through these networks. In the cases where the outcome does not depend on the previous phone, this simply means we are selecting the leftmost phone prediction displayed since it has the highest probability. When, however, it does

PHONEME	PHONE1	PHONE2	CONTEXT
hh	0.74 hh	0.15 hv	
ae	0.73 ae	0.19 eh	
d	0.51 jh	0.37 d	
y	0.90 y		(if d→d)
	0.84 -	0.16 y	(if d→jh)
uw	0.48 axr	0.29 er	

Figure 2 Pronunciation network for ‘had your’. Note the combination likely gives rise to the affricate [jh].

depend on the previous phone, then we are dealing with a transition probability and must find which of several possible paths is best. For example, the best result for Figure 2 is [hh ae jh axr].

4 SINGLE BEST PHONETIC PRONUNCIATION

We now turn to the evaluation of speech recognition performance using phonetic spellings of words in contrast with the phonemic spelling results described in Section 2.

Our first approach will be to expand each phonemic spelling into its most likely phonetic spelling. Thus, for example, the /t/ in **bottle** would be transcribed as [dɔ], its most likely allophone in this context (for American English).

In order to predict the most likely phonetic spelling of an utterance, we simply find the most likely path through its corresponding phonetic network as described in Section 3.6. With such a phonetic lexicon, having one pronunciation per word, we achieve 94.1% word accuracy on the Feb ’89 DARPA RM test set.

This is a 0.7% improvement over using the phonemic spelling. This modest improvement suggests that while we gain something by using a finer, more precise symbol set (and thus sharper acoustic models for it), we probably lose something by not always correctly guessing the appropriate phonetic transcription. We believe that usually in such cases, it is not that our technique incorrectly predicts the a priori most likely pronunciation, but that the speaker uses a less likely variant.

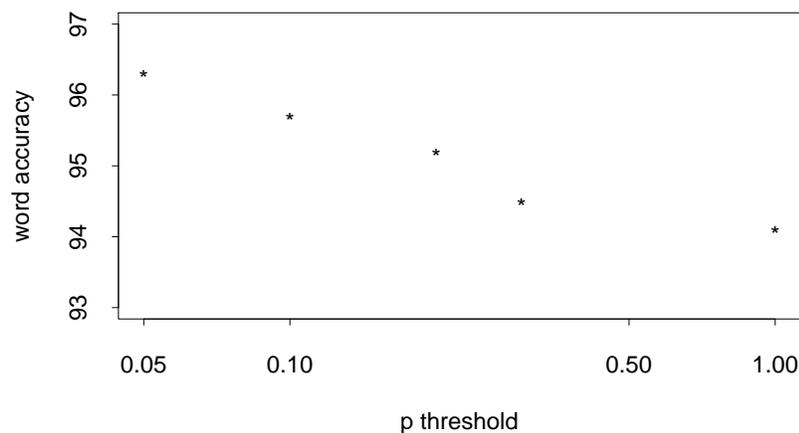


Figure 3 Word accuracy vs. probability threshold, p , on phone realization likelihoods for the DARPA RM Feb '89 test set

5 ALTERNATIVE PRONUNCIATIONS

Our next modification is to use multiple phonetic pronunciations per word. In order to predict the N most likely phonetic spellings we could simply find the N most likely paths through its corresponding phonetic network. We have, however, found it inconvenient to always have to expand such a network into distinct phonetic strings, but would sometimes rather use the transition network directly. Therefore, we instead limit the number of pronunciations alternatives by placing a threshold p on the minimum likelihood of a single phone realization. For example, in Figures 1 and 2, $p = 0.1$; there are no phone realizations with probability less than p . We add the proviso that we always keep the most likely phone realization for a given phoneme even if it is below the threshold so that we do not disconnect the network when p is large. For each value of p , we can consider all paths through the (truncated) pronunciation network to be the number of alternative pronunciations of that word/utterance. Note that when $p = 1.0$, we are left with the single, most likely pronunciation and as $p \rightarrow 0$, we get more and more pronunciation alternatives.

It is clear that unless we factor the realization likelihoods into our recognition score, that as $p \rightarrow 0$, very unlikely pronunciations with good acoustic match for incorrect words can degrade our recognition performance. To avoid this, we use Bayes Theorem to combine the acoustic and pronunciation alternative likelihoods into a single word/utterance likelihood [15, 12].

Figure 3 shows word accuracy vs. p for the DARPA RM Feb '89 test set using an alternative pronunciation lexicon with likelihoods. We see that performance improves as we increase the bushiness of the pronunciation networks. At the best value, $p = .05$, we get 96.3% word accuracy, 2.9% better than with phonemes alone.

We note with $p = .05$, if we were to expand the phonetic network for each word in the lexicon into distinct pronunciation strings, we would get on average about 17 pronunciations per word. The short words, of course, have relatively few pronunciations, while the longer ones considerably more.

We also note that the word accuracy of just the context-independent first pass is 95.1%. This remarkable result shows that even *context-independent* phone models considerably exceed *context-dependent* phoneme models in performance.

For the DARPA NAB Nov '94 test set we used $p = .05$ with the same model for predicting alternative phonetic realizations. This way we achieved 89.2% word accuracy, 1.9% better than when using phonemic representation.

We further extended this approach by utilizing the NAB training data to generate a more accurate model for predicting alternative phonetic realizations using the same techniques as described above.

Since we had available 37,000 training sentences (SI-284 training data set) for the NAB task, we could build a more accurate model of alternative realizations if accurate phonetic transcriptions for the training sentences were available. We generate the phonetic transcriptions of the training data by using the speech recognition system in the transcription mode with very bushy allophonic networks ($p = 0.001$). Although the automatic transcriber is not as accurate as the human transcribers used for creating the TIMIT database, we had twenty times as much data to train the model, as the NAB sentences are on average approximately twice as long as the TIMIT sentences. The decision tree was then re-trained on the automatic transcriptions.

When we tested the new NAB-based allophonic pronunciation networks with the same threshold $p = .05$ the word accuracy improved to 90.0%, which is 0.8% better than the TIMIT-based networks and 2.7% better than using the phonemic representation.

We conclude that substantial recognition performance improvements are possible when we use a probabilistic model for predicting phonetic realizations of phonemes.

REFERENCES

- [1] Breiman, L., et. al., *Classification and Regression Trees*, Monterey, CA: Wadsworth & Brooks, 1984.
- [2] Chen, F., "Identification of contextual factors for pronunciation networks," *Proc. ICASSP '90*, S14.9, 1990.
- [3] Coker, C. "A dictionary-intensive letter-to-sound program," *J. Acoust. Soc. Am.*, **78**, Suppl. 1, S7, 1985.
- [4] Coker, C., Church, K., Liberman, M., "Morphology and rhyming: two powerful alternatives to letter-to-sound rules," *Proc. ESCA Workshop on Speech Synthesis*, Autran, France, Sept. 1990.
- [5] Cohen, M., *Phonological structures for speech recognition*, U.C. Berkeley Ph.D thesis, 1989.
- [6] Fisher, W., Zue, V., Bernstein, D. and Pallet, D., "An acoustic-phonetic data base," *J. Acoust. Soc. Am.* **81**, Suppl. 1, 1987.
- [7] Giachin, E., Rosenberg, A., Lee, C., "Word juncture modeling using phonological rules for HMM-based continuous speech recognition," *Computer speech and language*, **5**, 1991.
- [8] Kruskal, J., "An overview of sequence comparison" , In *Time Warps, String Edits, and Macromolecules: the Theory and Practice of Sequence Comparison*, D. Sankoff and J. Kruskal, eds. , Reading, MA: Addison Wesley, pp. 1-44, 1983.
- [9] Ladefoged, P., *A Course in Phonetics.*, New York: Harcourt, Brace, and Jovanovich, 1982.

- [10] Lamel, L. and Gauvain, J., "Continuous speech recognition at LIMSI," *Final review of the DARPA ANNT Speech Program*, Palo Alto, CA, pp. 77-82, Sept. 1992.
- [11] Ljolje, A., "High accuracy phone recognition using context clustering and quasi-triphonic models," *Computer Speech and Language*, **8**, pp. 129-151, 1994.
- [12] Ljolje, A. and Riley, M.D., "Optimal speech recognition using phone recognition and lexical access," *ICLSP '92*, Banff, Canada, Oct. 1992.
- [13] Philips, M., Glass, J., Zue, V., "Modeling context dependency in acoustic-phonetic and lexical representations," *Proc. DARPA Speech and Natural Language Workshop*, pp. 71-76, Feb. 1991.
- [14] Riley, M., "Some applications of tree-based modeling to speech and language," *Proc. DARPA Speech and Natural Language Workshop*, Cape Cod, MA, pp. 339-352, Oct. 1989.
- [15] Riley, M.D. and Ljolje, A., "Lexical access with a statistically-derived phonetic network," *Eurospeech '91*, Genoa, Italy, Sept. 1991.
- [16] Riley, M.D., Ljolje, A., Hindle, D., and Pereira, F., "The AT&T 60,000 word speech-to-text system," *Eurospeech '95*, Madrid, Spain, Sept. 1995.
- [17] Randolph, M. "A data-driven method for discover and predicting allophonic variation," *Proc. ICASSP '90*, S14.10, 1990.
- [18] Shoup, J. "Phonological aspects of speech recognition," In *Trends in Speech Recognition*. W. Lea, ed, NY: Prentice-Hall, pp. 125-138, 1990.
- [19] Weintraub, M., Murveit, H., Cohen, M., Price, P., Bernstein, J., Baldwin, G., Bell, D., "Linguistic constraints in hidden Markov model based speech recognition," *Proc. ICASSP '89*, S13.2, 1990.