Corrective and Reinforcement Learning for Speaker-Independent Continuous Speech Recognition

Kai-Fu Lee and Sanjoy Mahajan

January 1, 1989
CMU-CS-89-100

Computer Science Department
Carnegie-Mellon University
Pittsburgh, PA 15213

Abstract

This paper addresses the issue of learning hidden Markov model (HMM) parameters for speaker-independent continuous speech recognition. Bahl *et al.* [Bahl 88a] introduced the *corrective* training algorithm for speaker-dependent isolated word recognition. Their algorithm attempted to improve the recognition accuracy on the training data. In this work, we extend this algorithm to speaker-*independent continuous* speech recognition. We use cross-validation to increase the effective training size. We also introduce a *near-miss sentence hypothesization* algorithm for continuous speech training. The combination of these two approaches resulted in over 20% error reductions both with and without grammar.

This research was sponsored by Defense Advanced Research Projects Agency Contract N00039-85-C-0163. The views and conclusions contained in this document are those of the author and should not be interpreted as representing the official policies, either expressed or implied, of the Defense Advanced Research Projects Agency, or the US Government.

Table of Contents

1. Introduction 1
2. Hidden Markov Models and Maximum Likelihood Training 3
3. The SPHINX Speech Recognition System 5
4. Corrective and Reinforcement Learning for Speaker-Independent Continuous Speech Recognition 6
   4.1 The IBM Corrective Learning Algorithm 6
   4.2 Using Cross Validation to Increase Training 7
   4.3 Reinforcement Learning for Continuous Speech Recognition 8
      4.3.1 Generating Near-Miss Phrase Substitutions 8
      4.3.2 Hypothesizing Near-Miss Sentences 12
   4.4 Final Algorithm Description 13
5. Results and Discussion 14
6. Conclusion 17
   Acknowledgements 18

1. Introduction

At present, hidden Markov models (HMMs) constitute the predominant approach to automatic speech recognition. HMM-based systems make certain structural assumptions, and then try to learn two sets of parameters—output probabilities, which represent speech events, and transition probabilities, which represent duration and timescale distortions. Most HMM-based systems use the Baum-Welch (or forward-backward) algorithm [Baum 72, Jelinek 76, Bahl 83], which adjusts the parameters to obtain an approximation to the maximum-likelihood estimates (MLE) of the HMM parameters.

Maximum likelihood estimators have many desirable properties, and many successful systems [Jelinek 85, Chow 87, Lee 89a, Rabiner 88] are based on MLE. However, maximum likelihood estimation has one serious flaw: it assumes that the underlying models are correct. In reality, however, typical HMMs make extremely inaccurate assumptions about the speech production process. This suggests two avenues of research: (1) attempt to rectify these assumptions, or (2) use new estimation techniques that work well in spite of these inaccurate assumptions. In this paper, we will consider the latter approach.

Bahl *et al.* [Bahl 88a] introduced the corrective training algorithm for HMMs as an alternative to the forward-backward algorithm. While the forward-backward algorithm attempts to increase the probability that the models generated the training data, corrective training attempts to maximize the recognition rate on the training data. This algorithm has two components: (1) *error-correction learning* — which improves correct words and suppresses misrecognized words, (2) *reinforcement learning* — which improves correct words and suppresses near-misses. Applied to the IBM speaker-dependent, isolated-word office correspondence task, this algorithm reduced the error rate by 16%.

In this study, we extend the corrective and reinforcement learning algorithm to *speaker-independent, continuous speech* recognition. Speaker independence presents some problems, because corrections appropriate for one speaker may be inappropriate for another. However, with a speaker-independent task, it is possible to collect and use a large training set. More training provides not only improved generalization but also a greater coverage of the vocabulary. We also propose the use of *cross-validation* to increase the effective training data size used to locate the misrecognitions needed by the correction algorithm. Cross-validation partitions the training data and determines misrecognitions using models trained on different partitions. This simulation of actual recognition leads to more realistic misrecognition hypotheses.

Extension to continuous speech is more problematic. With isolated-word input, both error-correcting and reinforcement training are relatively straighforward, since all errors are simple substitutions. Bahl, *et al.* [Bahl 88a] determined both misrecognized words and near-misses by matching the utterance against the entire vocabulary. However, with continuous speech, the errors include insertions and deletions. Moreover, many substitutions appear as phrase-

substitutions, such as *home any* for *how many*. In general, word boundaries are neither known nor reliably detectable. Without word-boundary information, it would be difficult to suppress misrecognized words and hypothesize near-misses. These problems make reinforcement learning difficult. We propose an algorithm that hypothesizes near-miss sentences for any given sentence. First, a dynamic programming algorithm produces an ordered list of likely *phrase substitutions*. Then, this list is used to hypothesize the near-miss sentences used in reinforcement learning.

We applied our corrective training procedure to the 997-word DARPA continuous resource management task, using the speaker-independent database. Without a grammar, we obtained a 20.3% error-rate reduction over the standard MLE-trained SPHINX System. With a word-pair grammar, we obtained a 23.4% reduction. These improvements are comparable to the IBM results with speaker-dependent, isolated-word recognition. Thus, we have successfully demonstrated the extensibility and applicability of the corrective training and reinforcement learning algorithm to speaker-independent continuous speech recognition.

In this paper, we first give a brief overview of hidden Markov models in Section 2. A brief description of the SPHINX system, on which this work is based, is presented in 3. We present our algorithm in Section 4, and our results in Section 5. Section 6 discusses possibilities for future work and finishes with a brief conclusion.

2. Hidden Markov Models and Maximum Likelihood Training

Hidden Markov models (HMM) were first described in the classic paper by Baum [Baum 72]. Shortly afterwards, they were extended to automatic speech recognition independently at CMU [Baker 75] and IBM [Bakis 76, Jelinek 76]. It was only in the past few years, however, that HMMs became the predominant approach to speech recognition, superseding dynamic time warping.

A hidden Markov model is a collection of states connected by transitions. Each transition carries two sets of probabilities: a transition probability, which provides the probability for taking this transition, and an output probability density function (pdf), which defines the conditional probability of emitting each output symbol from a finite alphabet, given that some transition from the state is taken. Figure 2-1 shows an example of a hidden Markov model with two output symbols, A and B.

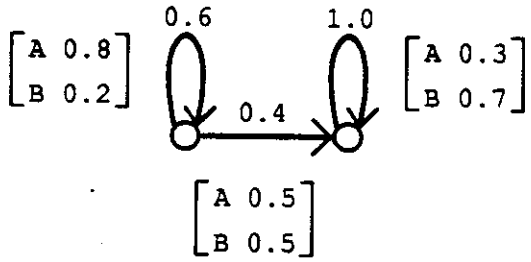

Figure 2-1: A simple hidden Markov model with two states, and two output symbols, A and B.

There are several types of hidden Markov models. In this study, we will assume discrete density HMMs, which are defined by:

- $\{s\}$—A set of states including an initial state S_I and a final state S_F.
- $\{a_{ij}\}$—A set of transitions where a_{ij} is the probability of taking a transition from state i to state j.
- $\{b_{ij}(k)\}$—The output probability matrix: the probability of emitting symbol k when taking a transition from state i to state j.

The forward-backward algorithm is used to estimate a and b. We provide only a simplistic sketch here; details of the algorithm can be found in [Bahl 83, Lee 88a]. The forward-backward algorithm adjusts a and b iteratively. For each iteration, the estimates from the previous iteration are used to count how frequently each symbol is observed for each transition, and how frequently each transition is taken from each state. These counts are then normalized into new parameters. Let $c_{ij}(k)$ represent the frequency (or count) that the symbol k is observed when the transition from i to j is taken, the new output probability $\overline{b}_{ij}(k)$ is given by the normalized

frequency:

$$\overline{b}_{ij}(k) = \frac{c_{ij}(k)}{\sum_{k'=1}^{K} c_{ij}(k')} \tag{1}$$

Similarly, transition probabilities are re-estimated by normalizing the frequency that a transition is taken from a particular state:

$$\overline{a}_{ij} = \frac{\sum_{k=1}^{K} c_{ij}(k)}{\sum_{\forall j'} \sum_{k=1}^{K} c_{ij'}(k')} \tag{2}$$

Baum [Baum 72] showed that re-estimating a and b, as shown in equations 1 and 2, will increase the likelihood of generating the training data, unless a local maximum has been reached. Although the forward-backward algorithm guarantees only a local maximum, it efficiently produces an approximation to the maximum-likelihood estimates (MLE) of the HMM parameters.

In spite of the many advantages of maximum-likelihood estimation, it suffers a serious problem, namely, it assumes that the underlying models, in this case HMMs, are correct [Brown 87]. However, HMMs are poor models of *real* speech, due mainly to the Markov independence assumption. With an incorrect model, there is no guarantee that maximum-likelihood estimation will converge to the best values for speech recognition.

3. The SPHINX Speech Recognition System

Our experiments in this paper were run by modifying an existing speech recognition system, SPHINX [Lee 89a]. SPHINX is a large-vocabulary, speaker-independent, continuous-speech recognition system based on maximum-likelihood HMMs.

SPHINX, which uses vector quantized LPC-derived cepstral coefficients in discrete HMM's, is based on phonetic hidden Markov modeling. Each word is represented by a pronunciation network of phones, and the set of sentences accepted by the grammar is represented by a network of words. Recognition in SPHINX is carried out by a Viterbi beam search [Viterbi 67, Schwartz 85]. While these techniques have worked well in speaker-dependent or isolated-word recognition, we have found that they alone are inadequate for our difficult task. It is necessary to improve these techniques to deal with *speaker independence* and *continuous speech*.

In order to deal with speaker independence, we experimented with various ways of adding knowledge to SPHINX. The simplest way to add knowledge to HMM's is to add more frame-based parameters. We use three sets of parameters : (1) instantaneous LPC cepstrum coefficients, (2) differenced LPC cepstrum coefficients, and (3) power and differenced power. These parameters are vector quantized separately into three codebooks, each with 256 entries. We found that quantizing these parameters separately both improved recognition accuracy and reduced VQ distortion. We also incorporated a word duration knowledge source into the recognizer.

Two great problems introduced by continuous speech are unclear function words and coarticulation. In order to improve the recognition of function words, we use *function-word-dependent phone models*, whose parameters depend on the word in which they appear. Since function words occur frequently, these models can be adequately trained. Moreover, since function words appear frequently in any task, they can be trained in a task-independent fashion. Finally, these models are still phone models, so we can interpolate their parameters with context-independent models when their training is insufficient. In order to deal with coarticulation, we introduce the *generalized triphone models*. Generalized triphone models are similar to context-dependent triphone models [Schwartz 85]. However, instead of modeling all left and right contexts, we use a maximum likelihood clustering procedure to merge similar contexts together. The use of function-word-dependent phone models and generalized triphone models gives us a total of 1076 models.

We applied SPHINX to the 997-word resource management task used by the DARPA projects. We used 4200 sentences produced by 105 speakers for training, and another 150 sentences by 15 different speakers for testing. We obtained word accuracies of 93.7%, and 70.6% for grammars with perplexity 60 and 997, respectively. More information about SPHINX can be found in [Lee 88a, Lee 88b, Lee 89a, Lee 89b, Lee 89c].

4. Corrective and Reinforcement Learning for Speaker-Independent Continuous Speech Recognition

4.1 The IBM Corrective Learning Algorithm

In view of the dependence of MLE on the problematic assumptions of HMMs, Bahl, *et al.* [Bahl 88a] proposed the IBM corrective training algorithm for speaker-dependent isolated-word recognition. This procedure, inspired by perceptron models [Minsky 69], attempts to tune the models to minimize recognition errors. This goal has a definite practical appeal, since error rate, not sentence likelihood, is the bottom line for speech recognition.

In order to minimize errors, the algorithm first attempts to recognize each training utterance u, representing some word w, with some initial model. If u is misrecognized as ω, then the parameters of the system are modified so as to make w more probable and ω less probable. This is the *corrective* component of the algorithm. The other component, which we call the *reinforcement* learning, is always activated whether or not u is recognized correctly. In that case, a list of *near-misses* ω_j are identified. Each near-miss is then made less probable with respect to the correct word. Figure 4-1 illustrates this algorithm in detail.

1. Generate an initial set of models from forward-backward training, preserving counts ($c_{ij}(k)$) for the transitions and output symbols.
2. For each training utterance u, use normalized c_{ij} to compute $P(u|w)$ for the correct word w, and $P(u|\omega_m)$ for a list of "confusable" words, ω_m. This list consists of:
   - misrecognitions — if $P(u|\omega_m) > P(u|w)$
   - near-misses — if $\log P(u|\omega_m) - \log P(u|w) > -\delta$

   where δ is a positive threshold determined *a priori*.
3. Run the forward-backward algorithm on each utterance u, using the model for the correct word w to obtain the counts c_{ij}^{w}. Do the same with the model for each ω_m to obtain the counts $c_{ij}^{\omega_m}$. Then, replace original counts c_{ij} with $c_{ij} + \gamma(c_{ij}^{w} - c_{ij}^{\omega_m})$. γ is an adjustment factor:
   - For misrecognitions, it is set to β.
   - For near misses, it decreases linearly from β to 0 as the difference in log-probabilities decreases from 0 to $-\delta$.
4. Replace any negative counts by a small positive constant, and continue with step 2, until enough iterations have been run.

Figure 4-1: The IBM corrective training algorithm.

Error correction occurs in step 3. By adding counts for the correct word (c_{ij}^{w}) to actual

counts for the models (c_{ij}), the correct word is made more probable. Conversely, by subtracting counts for the confusable words ($c_{ij}^{\omega_m}$) from c_{ij}, the confusable words are made less probable. The parameters of this algorithm include:

- δ — the threshold value for determining what constitutes a near-miss.
- β — the maximum step size in correction.

γ is directly computed from $\log P(u|w)$, $\log P(u|\omega_m)$, and β.

Since word boundaries are known in their isolated-word task, it was possible to generate ω_m, the list of near-miss words by matching the utterance with all the words in the vocabulary. Bahl, *et al.* actually used a fast match algorithm [Bahl 88b] which does this efficiently. From this list, misrecognitions and near-misses were determined using the criterion in step 2 of Figure 4-1.

Over a series of four experiments, the IBM corrective training algorithm produced an average error reduction of 16% on test data, and 88% on training data. These experiments successfully demonstrated the feasability of corrective training for speaker-dependent, isolated-word recognition.

The simplest way to apply IBM's corrective training algorithm to continuous speech is to treat each sentence as a "word." A misrecognition of a sentence can then be corrected by adjusting the counts for the entire sentence. This simple-minded approach has at least two flaws. First, there is no convenient way to produce multiple misrecognitions, so the *corrective* component would have at most one error per sentence. This provides very little training. Second, this approach does not suggest any good ways of generating near-misses, because there is no readily available list of near-miss sentences. This makes reinforcement learning impossible. In the next two sections, we will introduce techniques that solve both these problems.

4.2 Using Cross Validation to Increase Training

The IBM corrective training algorithm uses the same data to train the HMM probabilities and to determine what and how much to correct. However, recognition on training data invariably results in fewer and less realistic errors than does recognition on independent test data. SPHINX makes almost twice as many errors on a test set than on a training set. Thus, correcting on training data will provide only half as many misrecognitions for correction.

In order to alleviate this problem, we propose the use of *cross-validation*. First, the training data is divided into two partitions, and HMMs are trained on each partition. Then, HMMs trained from one partition are used to recognize the sentences from the other. Not only will we obtain many more errors this way, but these errors will also be more realistic. We then use these errors to correct the models trained on the entire set. Partitioning for cross-validation no longer makes sense after the first iteration, because the HMMs will have been trained on one partition and corrected on the other. Therefore, in our implementation we use the misrecognized

sentences from cross-validation for the first iteration. In future iterations, we reuse them as near-miss sentences.

4.3 Reinforcement Learning for Continuous Speech Recognition

Bahl, *et al.* [Bahl 88a] found that reinforcement (near-miss) learning aided the convergence of corrective training considerably. For isolated-word tasks, near-miss training is conceptually simple, since the only errors are simple word-for-word substitutions (such as *for* → *far*). To generate near-misses, Bahl, *et al.* used a list of near-miss words produced by a fast-match algorithm [Bahl 88b].

Such an approach is unsuitable for continuous speech, where we need to produce *near-miss sentences* given a correct sentence. This information is unavailable from a continuous speech recognizer due to pruning. We decompose this problem as follows:

1. Produce a long list of near-miss phrase substitutions, where each phrase may have zero to several words.
2. Use this list to hypothesize near-miss sentences by substituting one or more of the near-miss phrases with their respective replacements.

The next two sections will describe these two components of our reinforcement learning algorithm for continuous speech recognition.

4.3.1 Generating Near-Miss Phrase Substitutions

In this section, we describe our algorithm to generate a list of near-miss, or confusable, phrases. The first issue is the definition of a confusable phrase. It is inadequate to simply model word-for-word substitutions, because errors in continuous speech recognition are rarely so simple. More complex errors have been modeled by scoring routines [Pallett 87] that attempt to compute the error rate of a system, and provide a list of frequent errors. These routines typically consider insertions (*ship* → *the ship*) and deletions (*a sub* → *sub*) in addition to substitutions. While these three categories are adequate for determining the error rate of a system, they are unsuitable for finding near-miss phrases for two reasons. First, they contain no contextual information (*the* is more likely to be deleted in *list the uttered word* than in *the word was uttered*). Second, phrase substitutions (*during* → *that are in*, *how many* → *home any*) cannot be decomposed into substitutions, insertions, and deletions.

In view of the above, we model system errors as *near-miss phrase substitutions*. A near-miss phrase substitution is a pair of phrases, where each phrase may have zero or more words. We generate these confusable phrases as follows. First, we use cross-validation recognition to obtain realistic misrecognized sentences. Then, to find plausible phrase errors, each misrecognized sentence is matched against the corresponding correct one by a dynamic programming (DP) algorithm [Aho 74]. We could then define near-miss phrases as phrase alignments that have reasonable costs.

In order to align two sentences, each sentence must be first decomposed into a sequence of comparable units. For example, the scoring programs use words as units. The problem with using words as units is that they have no self-evident distance metric. Scoring programs use a simple distance metric, where if two identical words are aligned, the distance is zero, and substitutions, insertions, deletions are penalized with some constant distances. This type of distance metric is insensitive to similarities at the subword level, and will result in unreasonable alignments when multiple alignments are possible. Phones are a better unit, because multiple alignments can be resolved using phonetic similarity. However, there is no principled way of incorporating duration into phonetic distances. Also, similarities at the sub-phone level may be useful.

Therefore, we use the smallest unit available to us, and represent each sentence as a sequence of frames, where a frame is represented as an output distribution, b_{ij}. To obtain the distribution sequence, we align the actual utterance against the hidden Markov model for the correct sentence using the Viterbi algorithm. This is repeated for the misrecognized sentence. This process converts the correct and misrecognized sentences to their corresponding $\{b_{ij}\}$ sequences.

In order to align two b_{ij} sequences, we need a distance metric between the b_{ij}'s. We use an information theoretic distance that measures the change in entropy when the two distributions are merged [Lee 88a]. These distances are scaled to lie in [0,1], with a cost of 1 for insertions and deletions.

Given the b_{ij} sequences for the correct sentence (C) and for the misrecognized sentence (M), each with length L, we are now in a position to generate near-miss phrase substitutions. A phrase substitution can be thought of as a "box", indicated by its coordinates, C_i, C_j, M_k, M_l. This "box" matches two phrases: (1) frames i to j of the correct b_{ij} sequence, and (2) frames k to l of the misrecognized b_{ij} sequence. The cost of a box can be determined by aligning the two sequences of b_{ij}'s using a dynamic programming (DP) algorithm [Aho 74] to find the optimal alignment and cost. This cost is defined by the following equations:

$$Cost(C_i, C_i, M_k, M_k) = 0 \tag{3}$$

$$Cost(C_i, C_j, M_k, M_l) = MIN \begin{cases} Cost(C_i, C_{j-1}, M_k, M_{l-1}) + Dist(C_j, M_l) \\ Cost(C_i, C_{j-1}, M_k, M_l) + 1 \\ Cost(C_i, C_j, M_k, M_{l-1}) + 1 \end{cases} \tag{4}$$

where $Dist(C_j, M_l)$ is the entropic distance between frame j of the correct sequence and frame l of the misrecognized sequence.

For example, the box in Figure 4-2(a) shows alignment of two entire sentences, which gives us a *globally optimal cost*, or $Cost(C_1, C_L, M_1, M_L)$. Figure 4-2(b) shows the decomposition of the global box into three boxes. The central box represents the substitution *who is in* → *will wasn+t*. We consider the central box a near-miss phrase substitution if *the total cost of the three*

boxes is within ε of the globally optimal cost (that of the box in Figure 4-2(a)). In other words, the phrase substitution designated by the central box will be considered a near-miss if the following equation is satisfied:

$$Cost(C_1, C_{i-1}, M_1, M_{k-1}) + Cost(C_i, C_j, M_k, M_l) + Cost(C_{j+1}, C_L, M_{k+1}, M_L) \leq Cost(C_1, C_L, M_1, M_L) + \varepsilon \quad (5)$$

This gives us a principled way of finding good near-miss phrases that actually caused the misrecognition. In the example, the central box in Figure 4-2(b) was considered a near-miss, while the central box in Figure 4-2(c) was not. All the near-miss phrase substitutions are saved, along with their cost in the central box, for later use.

The costs of all possible peripheral boxes can be precomputed efficiently, because there are only $C_W \times M_W$ possible end points for the left box, and $C_W \times M_W$ possible begin points for the right box (C_W is the number of words in the correct string, and M_W is that in the misrecognized string). As a byproduct of the DP algorithm, alignment of the entire sentences yields all the possible costs for left boxes. Similarly, aligning the sentences in reverse yields the costs for the possible right boxes. However, there are almost $(C_W \times M_W)^2$ possible central boxes, one for every possible combination of the two peripheral boxes, and DP alignment will have to be run for each combination. Some pruning is needed to contain the central box computation. We use branch-and-bound to discard any box for which the combined peripheral cost already exceed the globally optimal cost by more than ε, or:

$$Cost(C_1, C_{i-1}, M_1, M_{k-1}) + Cost(C_{j+1}, C_L, M_{k+1}, M_L) > Cost(C_1, C_L, M_1, M_L) + \varepsilon \quad . \quad (6)$$

In addition, we restrict the substitution phrases to at most three words. We felt that longer phrases would become too specific for the training data.

We processed 4150 pairs of correct and recognized sentences using the above algorithm, obtaining a list of 13000 phrase substitutions in about 4 hours of CPU time on a Sun-4. As an example of the substitutions produced, the substitutions produced for matching "*Who is in west siberian sea?*" with "*If will wasn't last siberian sea*" are:

west → *last*
is in → *wasn't*
who → *will*
who is in → *will wasn't*

To digress momentarily, we would like to point out that the algorithm described in the section could be modified into a sophisticated scoring algorithm that can give more accurate estimates of error rate and automatically provide a list of errors of analysis [Pallett 87, Hunt 88].

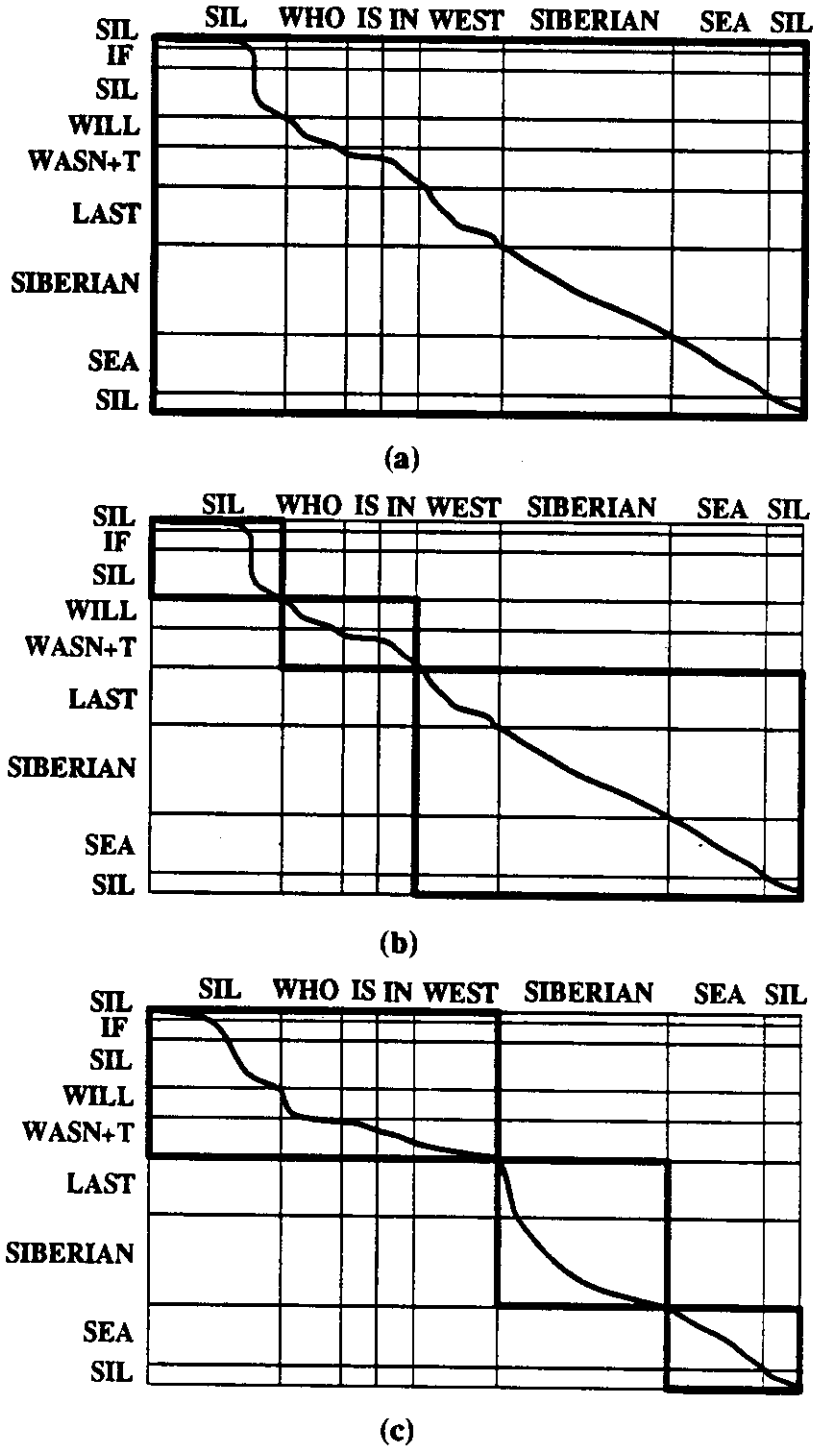

Figure 4-2: Three examples of DP alignment in the near-miss phrase generation. (a) is the globally optimal path, (b) is an example of a good alignment, and (c) is an example of a poor alignment. The quality of the alignment is assessed by comparing the resulting distance with that of the optimal path.

4.3.2 Hypothesizing Near-Miss Sentences

The candidate list produced by the phrase generation algorithm is then used by the *sentence hypothesizer*, which heuristically hypothesizes likely near-miss sentences. Almost any reasonable method will do, but for completeness we describe our algorithm in Figure 4-3. This algorithm hypothesizes one near-miss sentence given a correct sentence. Since it is non-deterministic, we can iterate it until we have enough near-misses for each correct sentence. We use an average of 6 near-miss sentences per original sentence.

1. Start at the beginning of the correct sentence by setting the current word position to zero.
2. Make a list of possible phrase substitutions starting at the current position.
3. Randomly make a substitution from the list determined in step 2. The probability of making a substitution is a monotonically decreasing function of the cost in the DP process. Making no substitution is allowed, with no cost.
4. If we made a substitution, advance the current position to the end of the substituted phrase. Otherwise, advance the current position by one word.
5. If at end of sentence, stop. Otherwise, go to step 2.

Figure 4-3: The near-miss sentence hypothesization algorithm.

This algorithm is very efficient. We can hypothesize 250 near-miss sentences per second on a Sun-4. The hypotheses generated by the algorithm depend greatly on the phrase substitution list with which it is provided. Using the substitution list derived from recognition without grammar, the hypotheses produced for *who is in west siberian sea* include:

who is in were sub area be
who is in west it siberian be
who is in when siberian same
who is in when siberian it sea

On the other hand, using a word-pair-grammar substitution list, we obtained a different substitution list:

who was the west siberian sea
who is in the west siberian sea
who list at west siberian sea
who has been in west siberian sea

Without a grammar, many ungrammatical substitutions occur, such as *west* → *when* and *sea* → *be*. But with a grammar, the hypothesized sentences are more grammatical, with substitutions such as *is* → *was* and *in* → *at*.

4.4 Final Algorithm Description

Figure 4-4 summarizes our corrective and reinforcement algorithm for continuous speech recognition. There is one implementational detail that we have not covered. A problem we encountered during recognition was that corrective training would make some probabilities too small for test data. As a result, a few sentences could not be recognized. To remedy this problem, we use a large β and then smooth the trained parameters with the MLE parameters. For example, with a smoothing weight of 0.2 for the MLE parameters and 0.8 for the new parameters, we ensure that no parameter can shrink by more than 80% from its MLE estimate. This is more sensitive than simply using a smaller β, or setting a large minimum value for output probabilities.

1. Partition the training sentences into two equal sections for cross-validation and train a set of models on each half.
2. Recognize each half with the models trained on the other half (cross-validation).
3. Perform the DP algorithm on misrecognized sentences in Section 4.3.1 to obtain a list of near-miss phrase substitutions.
4. Iterate the non-deterministic algorithm described in Figure 4-3 to hypothesize a list of near-miss sentences.
5. Generate the list of confusable sentences (ω_m) by combining:
   - The actual misrecognitions from the models of the previous iteration.
   - The list of hypothesized near-miss sentences in the previous step.
   - Sentences from cross-validation, for iterations after the first.
6. Run the forward-backward algorithm on each spoken sentence u, using the model for the correct sentence w to obtain the counts c_{ij}^{uw}. Do the same with each ω_m to obtain the counts $c_{ij}^{u\omega_m}$. Then, replace the each original count c_{ij} with $c_{ij} + \gamma(c_{ij}^{uw} - c_{ij}^{u\omega_m})$. γ is an adjustment factor:
   - For misrecognitions, it is set to β.
   - For near misses, it decreases linearly from β to 0 as the difference between $\log P(u|w)$ and $\log P(u|\omega_m)$ decreases from 0 to $-\delta$.
7. Smooth the resulting models with the MLE parameters, using a weighted average.
8. Go to step 3, until a sufficient number of iterations have been run.

Figure 4-4: The final corrective and reinforcement training algorithm.

5. Results and Discussion

The algorithm described in Section 4.4 can be used for recognition systems with and without grammar. We applied the algorithm to both cases.

To obtain the baseline MLE results, we trained the SPHINX system using the forward-backward algorithm, as described in Chapter 3. We applied SPHINX to the 997-word TI Resource Management task [Price 88], using 4150 training sentences (104 speakers) and 150 test sentences (10 speakers). Without a grammar, the perplexity is 997. With the word-pair grammar, which knows only about the legality of pairs of words classes, the perplexity is 60.

Our first experiment was run on the no grammar recognizer, where the MLE models made 375 errors[1] on test data, for an error rate of 29.4%. One iteration of simplistic corrective training, without near-misses or cross-validation, reduces the number of errors on test data to 336. More realistic cross-validation sentences reduces this to 329. Finally, one iteration of the complete algorithm, described in Figure 4-4, with reinforcement learning reduces this number to 316. Thus, after one iteration, our corrective training algorithm achieves a 15.7% error rate reduction on test data. Most of the improvement comes from basic corrective training, although a significant portion comes from enhancements such as cross-validation and reinforcement learning.

System Configuration	Errors on Test Set	% Test Set Error Reduction
MLE Training	375 (29.4%)	--
Corrective only	336 (26.3%)	10.4%
+ Cross-validation	329 (25.8%)	12.3%
+ Reinforcement	316 (24.7%)	15.7%

Table 5-1: Results of maximum-likelihood training vs. *one iteration* of variants of corrective and reinforcement training.

We then ran additional iterations of the corrective and reinforcement learning algorithm. For each iteration, we used the system from the previous iteration to produce misrecognitions for corrective training. A different set of near-miss sentence hypotheses was used, without regenerating the near-miss phrase substitutions. In addition, the misrecognitions from cross-validation were repeatedly used. The results are shown in Table 5-2. Also shown are the error rates on the training sentences. Note that the results shown for MLE training were obtained

[1] An error could be a substitution, a deletion, or an insertion.

without cross-validation[2], so that the entries in the first column would be more comparable. We found that after running two iterations, the result improved only negligibly. The final result gives an error rate of 23.4%, for an error rate reduction of 20.3% over the MLE models.

	Errors on Training Set	Errors on Test Set	% Test Set Error Reduction
Iteration 0 (MLE)	6122 (18.8%)	375 (29.4%)	-----
Iteration 1	4008 (10.9%)	316 (24.7%)	15.7%
Iteration 2	2218 (6.8%)	302 (23.6%)	19.5%
Iteration 3	1896 (5.2%)	299 (23.4%)	20.3%

Table 5-2: Results for iterative training of corrective and reinforcement training without grammar.

We also applied the corrective training algorithm to the models used by SPHINX's word-pair grammar. We first tried to use the models adjusted with the no-grammar hypotheses, but the error rate was actually greater than with MLE models. This happens because the two grammars make very different errors, which we verified by comparing their near-miss phrase hypotheses. For example, the word-pair grammar recognizer usually does not confuse *than* with *their*, but the no grammar one often does. Adjusting the model parameters that disambiguate these two words might actually hurt the word-pair recognition rate. So, we regenerated the cross-validation data and the phrase substitution list for the word-pair grammar correction. The results are shown in Table 5-3. As expected, comparable error reduction with the no grammar recognizer was achieved.

	Errors on Training Set	Errors on Test Set	% Test Set Error Reduction
Iteration 0 (MLE)	1799 (4.9%)	81 (6.3%)	-----
Iteration 1	809 (2.2%)	68 (5.3%)	16.0%
Iteration 2	689 (1.9%)	64 (5.0%)	21.0%
Iteration 3	650 (1.8%)	62 (4.9%)	23.4%

Table 5-3: Results for iterative training of corrective and reinforcement training with a word-pair grammar.

IBM reported error rate reductions of 16% and 88% on test data and training data respectively [Bahl 88a]. With more training, we report 20% and 72% without grammar; 23%

[2]With cross-validation, there are about 20-30% more errors than the test set, because it is a test set evaluation with less training data.

and 63% with grammar. Thus, we have not only demonstrated the extensibility of the corrective training algorithm to speaker-independent continuous speech recognition, but also narrowed the gap between training and testing results through the use of more training and cross validation.

6. Conclusion

Hidden Markov models with maximum-likelihood estimation constitute the predominant approach to automatic speech recognition today. However, maximum likelihood produces inferior results when the underlying models are incorrect, which HMM's obviously are as models of real speech. For this reason, the IBM corrective training algorithm produced good results when applied to the IBM isolated-word, speaker-dependent, office-correspondence task. The basic idea of this algorithm is to modify the HMM parameters so as to maximize the recognition rate on the training set. This is accomplished by making the correct words more probable, and the confusable words less probable. This paper extended this algorithm to continuous, speaker-independent speech recognition.

In order to increase the effective training size, we used cross-validation. In order to extend the algorithm to continuous speech, we introduced an algorithm that hypothesized near-miss sentences. This algorithm has two components: (1) a near-miss phrase substitution generator that used a dynamic programming algorithm to produce a long list of possible phrase substitutions, and (2) a non-deterministic near-miss sentence hypothesizer that used the phrase substitution list to hypothesize possible near-miss sentences from a correct sentence. These enhancements, aided by the use of a large training database, led to error rate reductions of 20.3% without grammar, and 23.4% with a word-pair grammar. One notable finding was that grammar-specific training was necessary, because corrections appropriate for one grammar may be suboptimal, or even harmful, to another grammar. Another finding was that smoothing a heavily-corrected set of parameters with MLE parameters led to the best results.

These results clearly demonstrated that the corrective training algorithm is applicable to speaker-independent, continuous speech recognition. The applicability of this algorithm to continuous speech is demonstrated through the use of novel algorithms for near-miss sentence hypothesization. The applicability of this algorithm to speaker-independent recognition is also important, because much more training data can be collected in speaker-independent mode. Through the use of a large multi-speaker database and cross-validation to increase the effective training size, we have narrowed the gap between the results of training and testing data. However, there is still a large difference between training and testing results. In order to further reduce this difference, more training will be needed, and more efficient techniques must be investigated to deal with the increased training.

There are number of other directions for future work. The phrase generation method described in this paper relied on having a large training database. This is practical for a 1000-word vocabulary, but would not be for a 20,000 word system. A method to generate confusable phrases given only a grammar and a dictionary would be very useful.

Another research area is whether the corrective training algorithm will be useful when training data is sparse. For example, could we improve a model even when it has not been observed? Positive results in this area will make corrective training an excellent speaker-

adaptation algorithm. Otherwise, it will still be a good adaptation algorithm when a reasonable amount of speaker-specific data is available.

Corrective and reinforcement learning is but one innovation that departs from the traditional maximum-likelihood estimation. A few other techniques, such as maximum mutual information estimation [Brown 87] and minimum discrimination information estimation [Ephraim 88], have been proposed. We believe that this is a rich area for further research.

Acknowledgements

The authors would like to thank Hsiao-Wuen Hon and Joe Keane for discussions and help. We would also like to thank DARPA for the support.

References

[Aho 74] Aho, A., Hopcroft, J., Ullman, J.
The Design and Analysis of Computer Algorithms.
Addison-Wesley Publishing Company, 1974.

[Bahl 83] Bahl, L. R., Jelinek, F., Mercer, R.
A Maximum Likelihood Approach to Continuous Speech Recognition.
IEEE Transactions on Pattern Analysis and Machine Intelligence
PAMI-5(2):179-190, March, 1983.

[Bahl 88a] Bahl. L.R., Brown, P.F., De Souza, P.V., Mercer, R.L.
A New Algorithm for the Estimation of Hidden Markov Model Parameters.
In *IEEE International Conference on Acoustics, Speech, and Signal Processing*. April, 1988.

[Bahl 88b] Bahl. L.R., Brown, P.F., De Souza, P.V., Mercer, R.L.
Obtaining Candidate Words by Polling in a Large Vocabulary Speech Recognition System.
In *IEEE International Conference on Acoustics, Speech, and Signal Processing*. April, 1988.

[Baker 75] Baker, J. K.
The DRAGON System -- An Overview.
IEEE Transactions on Acoustics, Speech, and Signal Processing
ASSP-23(1):24-29, February, 1975.

[Bakis 76] Bakis, R.
Continuous Speech Recognition via Centisecond Acoustic States.
In *91st Meeting of the Acoustical Society of America.* April, 1976.

[Baum 72] Baum, L. E.
An Inequality and Associated Maximization Technique in Statistical Estimation of Probabilistic Functions of Markov Processes.
Inequalities 3:1-8, 1972.

[Brown 87] Brown, P.
The Acoustic-Modeling Problem in Automatic Speech Recognition.
PhD thesis, Computer Science Department, Carnegie Mellon University, May, 1987.

[Chow 87] Chow, Y.L., Dunham, M.O., Kimball, O.A., Krasner, M.A., Kubala, G.F., Makhoul, J., Roucos, S., Schwartz, R.M.
BYBLOS: The BBN Continuous Speech Recognition System.
In *IEEE International Conference on Acoustics, Speech, and Signal Processing*, pages 89-92. April, 1987.

[Ephraim 88] Ephraim, Y., Rabiner, L.
On the Relations Between Modeling Approaches and Information Sources.
In *IEEE International Conference on Acoustics, Speech, and Signal Processing*, pages 24-27. April, 1988.

[Hunt 88] Hunt, M.
Evaluating the Performance of Connected-Word Speech Recognition Systems.
In *IEEE International Conference on Acoustics, Speech, and Signal Processing*, pages 457-461. April, 1988.

[Jelinek 76] Jelinek, F.
Continuous Speech Recognition by Statistical Methods.
Proceedings of the IEEE 64(4):532-556, April, 1976.

[Jelinek 85] Jelinek, et al.
A Real-Time, Isolated-Word, Speech Recognition System for Dictation Transcription.
In *IEEE International Conference on Acoustics, Speech, and Signal Processing*. March, 1985.

[Lee 88a] Lee, K.F.
Large-Vocabulary Speaker-Independent Continuous Speech Recognition: The SPHINX System.
PhD thesis, Computer Science Department, Carnegie Mellon University, April, 1988.

[Lee 88b] Lee, K.F., Hon, H.W.
Large-Vocabulary Speaker-Independent Continuous Speech Recognition.
In *IEEE International Conference on Acoustics, Speech, and Signal Processing*. April, 1988.

[Lee 89a] Lee, K.F., Hon. H.W., Reddy, R.
An Overview of the SPHINX Speech Recognition System.
IEEE Transactions on Acoustics, Speech, and Signal Processing , To Appear in, 1989.

[Lee 89b] Lee, K.F.
Automatic Speech Recognition: The Development of the SPHINX System.
Kluwer Academic Publishers, Boston, 1989.

[Lee 89c] Lee. K.F., Hon, H.W.
The SPHINX Speech Recognition System.
In *IEEE International Conference on Acoustics, Speech, and Signal Processing*. April, 1989.

[Minsky 69] Minsky, M., Papert, S.
Perceptrons.
MIT Press, Cambridge, Massachusetts, 1969.

[Pallett 87] Pallett, D.
Test Procedures for the March 1987 DARPA Benchmark Tests.
In *DARPA Speech Recognition Workshop*, pages 75-78. March, 1987.

[Price 88] Price, P.J., Fisher, W., Bernstein, J., Pallett, D.
A Database for Continuous Speech Recognition in a 1000-Word Domain.
In *IEEE International Conference on Acoustics, Speech, and Signal Processing*. April, 1988.

[Rabiner 88] Rabiner, L.R., Wilpon, J.G., Soong, F.K.
High Performance Connected Digit Recognition Using Hidden Markov Models.
In *IEEE International Conference on Acoustics, Speech, and Signal Processing*. April, 1988.

[Schwartz 85] Schwartz, R., Chow, Y., Kimball, O., Roucos, S., Krasner, M., Makhoul, J.
Context-Dependent Modeling for Acoustic-Phonetic Recognition of Continuous Speech.
In *IEEE International Conference on Acoustics, Speech, and Signal Processing*. April, 1985.

[Viterbi 67] Viterbi, A. J.
Error Bounds for Convolutional Codes and an Asymptotically Optimum Decoding Algorithm.
IEEE Transactions on Information Theory IT-13(2):260-269, April, 1967.