9

# LARGE VOCABULARY ISOLATED WORD RECOGNITION

**Vishwa Gupta**
**Matthew Lennig**

*Bell-Northern Research
and INRS-Télécommunications,
Montreal, Quebec, Canada, H3E 1H6*

## 1 INTRODUCTION

Many applications require recognition of spoken isolated words or phrases from a large vocabulary. For example, the goal of the 86000-word recognizer at INRS-Télécommunications [14] is to transcribe speech spoken as a sequence of isolated words. The sentences to be read are chosen arbitrarily from a variety of sources, including newspapers, books, magazines, etc. Another example is the StockTalk system running at BNR Montreal [24], which dispenses real time stock quotes by voice over the telephone for stocks traded in New York, Toronto and NASDAQ stock exchanges. The vocabulary for this system consists of words or phrases spoken in isolation. This system requires speaker-independent recognition over the telephone, while the first example requires speaker-dependent recognition over high quality microphones.

A typical flowchart for recognizing words or phrases in such applications is shown in Figure 1. There are many variations to this flow chart depending on the preferences of individual researchers, and many possibilities within each block. In describing these blocks, we will be guided by our own experiences, and to some extent will reflect our point of view.

At a higher level, we can divide the speech recognition algorithms into two categories: word-based or sub-word-based. Word-based recognition algorithms create a model for each word or phrase in the vocabulary. These algorithms are only suitable for small vocabularies. Each word-model has to be trained from several utterances of the word. For large vocabulary, word-based recognition requires unreasonable computing capabilities and a large effort in data collection in order to train the word models.

The alternative is to use sub-word based recognizers. The sub-words could correspond to phonemes, diphones, demi-syllables, speech segments etc. The idea in all these units is to reduce the total number of models required to transcribe any possible word in the vocabulary. Phonemes result in the smallest number of models. In phoneme based recognizers, each word is represented as a sequence of phonemes. Since there are roughly 40 phonemes in English, we can get away with training only 40 models instead of thousands for word-based systems. For this reason, phoneme-based recognizers require much less speech data to train the models. The computing complexity is also significantly less than that for word-based recognizers. In fact, the point where the computing for word-based recognizers becomes unreasonable is around a few hundred words.

Even though we are talking about models for phonemes, many researchers have shown that creating models for allophones [23] improves recognition accuracy. Allophones are phonemes in context. The acoustic realization of phonemes varies considerably depending on context (previous and following phonemes), on stress, on position in the syllable, on dialect etc. By representing the major variations as separate models, we can improve recognition accuracy significantly. However, there is some debate on the optimal number of allophones required to achieve good recognition accuracy. Various research groups have used anywhere from a few hundred to a few thousand allophones. Also, how to create these allophones is an issue. For example, Kai-Fu Lee [23] and Hon [15] have used phonemes in triphone contexts (previous and following phoneme) and a measure based on entropy reduction to choose their triphones. Bahl et al [6] at IBM have been recommending an approach based on data driven decision trees, and their allophone contexts can span up to 3 phonemes to the left and three phonemes to the right. The contextual variations due to co-articulation are a bigger problem in continuous speech than in words spokens in isolation, and Gopalakrishnan has probably addressed this issue in detail in the Chapter on continuous speech recognition.

Phonemes or allophones are not the only units used successfully in speech recognition. Bahl et al [5] use fenones as basic units. Fenones are data driven, and not driven by linguistic theory. A word in the vocabulary is represented as a sequence of fenones. Approximately 200 fenones are used to represent any word in the vocabulary. Fenones are probably described in detail in the article by Gopalakrishnan, since the speech recognition group at IBM has championed this approach. We will primarily stick to phonemes or allophones in our discussions. The search techniques or language models do not depend on the choice of phonemes or fenones, and all the algorithms which apply to phonemes also apply to other units as well.

Even though there are many ways to create phoneme or allophone models, the current best methods use Hidden Markov Models (HMMs). Many different styles of models have been used by various researchers. We can classify these styles into three broad categories based on the probability density function used for the observation vectors: discrete (or VQ-based) models, Gaussian mixture density based models, and tied mixture models. In VQ-based models, the observation vectors are mapped into discrete units by using one or more codebooks. For example, in Gupta et al [13], two separate codebooks are used to represent static and dynamic feature vectors. Kai-Fu Lee [23] uses three codebooks to represent the feature vector: one for the static parameters, one for the dynamic parameters, and one for the power and the dynamic power. Juang et al [18] use Gaussian mixture densities to represent the probability distribution of the observation vector. Tied mixture models also correspond to the Gaussian mixture models where the Gaussian mixtures are shared by all the transitions in one or more models.

Many different training algorithms have been used to train these models. These training algorithms include Viterbi training, forward-backward training, maximum mutual information (MMI) training, corrective training etc. In this Chapter we will focus on search algorithms, detailed matching strategies, and language modeling.

## 2 SEARCH ALGORITHMS

The goal of the recognizer is to transcribe words spoken as a sequence of isolated words. Since the words are spoken in isolated fashion, we can find the endpoints of each word, and transcribe each word separately. The task of the search algorithm is to find the most likely word hypothesis or to narrow this hypothesis to a few possible choices. Computing considerations dictate that we accomplish this with as little computing as possible without losing the correct choice from the hypothesis list.

A simplistic approach to large vocabulary recognition would compare the spoken word against all possible phonemic transcriptions for all the words in the vocabulary. To score all the words in the vocabulary in real-time would require hardware with computing capability on the order of hundreds of millions of floating point operations per second. The cost of the hardware to provide such computing capabilities would be prohibitive. To reduce the number of candidates for detailed comparison, we require an algorithm to perform fast

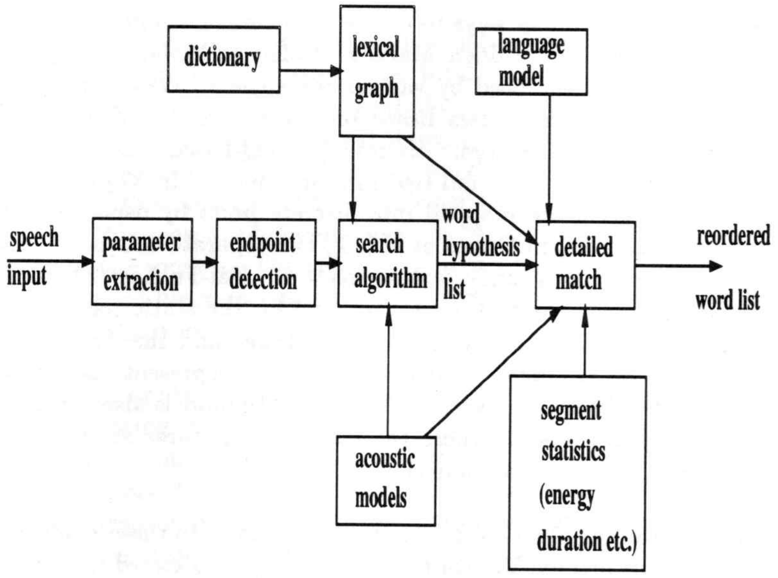

**Figure 1** Block diagram for a typical large vocabulary isolated word speech recognition system.

search, reducing significantly the size of the list for subsequent detailed matching. A number of algorithms exist which reduce this search. These search algorithms can be classified into three general categories: search algorithms based on Viterbi decoding, those based on $A^*$ heuristic search, and those based on fast match search.

## 2.1 Search Algorithms based on Viterbi decoding

The Viterbi decoding algorithm [31] provides an efficient way to get the most likely candidate. The earliest application of Viterbi decoding to speech recognition dates back to Vintsyuk [30]. The Viterbi algorithm provides the optimal frame synchronous search through a lexical graph whose branches represent allophonic or phonemic HMMs, and any path through the graph represents a lexical entry in the dictionary. Lee and Rabiner [22] give a good tutorial on the frame-synchronous search algorithms. Let us look at a rough estimate of computing involved in this frame-synchronous search algorithm. The computing complexity is proportional to the total number of transitions in all the HMMs associated with the lexical graph. For example, let us assume that the graph for a dictionary containing 20,000 phonemic transcriptions has a total of 100,000 branches. If each HMM has an average of 15 transitions, then the total computing per frame is proportional to 1,500,000. For a frame rate of 100 frames/second, we require computing capability in the order of 1500 million operations per second.

Another disadvantage of Viterbi algorithm is that it does not give us multiple choices economically. To generate multiple choices, we have to maintain multiple choices at each node in the lexical graph [22] resulting in a corresponding increase in computing. Soong [29] has shown how multiple choices can be obtained by carrying out a tree-trellis search after the Viterbi algorithm. His algorithm requires same computing complexity as the Viterbi algorithm for generating the top choice, but requires more memory for storing the intermediate values during Viterbi search. These values are then used during the tree-trellis search.

In order to reduce the computing even further, some people have resorted to beam search. Viterbi algorithm maintains all possible paths and extends these paths every frame, and the most likely path at the last frame corresponds to the optimal path. By extending only the most promising paths, the overall computing can be reduced significantly. For example, Ney [25] has shown that he can reduce the computing by an order of magnitude by using beam search together with proper structures to identify active partial paths. The beam search requires additional memory in order to track which paths are still alive and need to be extended during Viterbi search.

The tuning parameters which control beamwidth depend on the recognition task, and are usually set so as to minimize decoding errors while reducing computing as much as possible. These parameters may again have to be tuned even for the same task if the models or vocabulary or both are changed. If the graph represents a tree, then the beam search can also give multiple hypotheses. However, generating multiple hypotheses would require significant increase in computing, since the beam width would have to be increased significantly. To reduce the beamwidth, language model and any other sources of knowledge are incorporated during the frame-synchronous beam search [22] [25].

## 2.2 Search Algorithms based on $A^*$ algorithm

$A^*$ search algorithm [26] is a best-first optimal heuristic search as long as all its conditions are satisfied. The search keeps the partial paths on a stack and extends the partial path with the highest likelihood. The likelihood of the partial path includes the actual likelihood of the partial path and an estimate of the remaining path. The heuristic function which estimates the likelihood of the remaining path must provide a likelihood greater than or equal to the actual likelihood of the remaining path. As long as this condition is satisfied, $A^*$ search ensures optimality. There are some trivial heuristic functions which satisfy the optimality condition (for example, likelihood of 1). However, such functions would lead to a slow breadth first search. The heuristic functions which yield fast and focussed search are the ones which estimate the likelihood of the remaining path close to the actual likelihood.

An early example of the $A^*$ search algorithm is the stack decoding algorithm [16] [17]. In this algorithm, the likelihood of the remaining path was estimated as the likelihood of observing the remaining acoustic data independent of the path by collecting long term acoustic statistics. This likelihood was multiplied by a factor to ensure optimality. To avoid the search becoming breadth first search, the factor was controlled in order to ensure that the longest paths were slowly favored and extended.

Another example of the $A^*$ search algorithm is the syllabic graph search algorithm used by Gupta, Lennig and Mermelstein [12]. In this algorithm, they first find the number of syllables $n$ in the word using syllable models, and then carry out an $A^*$ search through a network representing n syllables. This syllable network is much smaller than the network representing the entire lexicon (a few thousand branches versus a few hundred thousand branches). Every branch of the graph is scored independent of its phonetic context, allowing a fast computation of the branch likelihoods. The $A^*$ search then generates multiple choices using these branch likelihoods. In this search strategy, the scoring of branch likelihoods is fast. The speed of search depends on number of choices necessary in order to include the correct choice among these hypotheses. For the 86,000 word recognizer, this algorithm required on an average 300 choices per word to keep the number of search errors to less than 2% (search error is when the correct hypothesis is not included in the hypothesis list).

Another example of $A^*$ search is the $A^*$-admissible lexical search used by Kenny et al [20]. In this search algorithm, Kenny defines a class of heuristic functions which allow for a tradeoff between the efficiency of the search and the burden of evaluating the heuristic scores. The heuristic scores for the partial path are estimated using a reduced graph together with the Viterbi scoring algorithm. Two examples of these reduced graphs are two-phone and one-phone look ahead graphs. In two-phone look ahead graph, the reduced graph only ensures that the next two phones are identical to those in the lexical tree. This allows a much simpler graph to be constructed to estimate the heuristic scores. The heuristic scores by design turn out to be higher than the Viterbi scores. The two scores are merged by considering all possible segmentations corresponding to exact and heuristic scores. Only a few of these segmentation hypotheses have to be saved in order to achieve optimal decoding, resulting in significant savings in computing.

The advantage of this algorithm is that the actual score at the end of the search corresponds to Viterbi likelihood, and therefore, the search can be stopped as soon as the necessary n hypotheses have been obtained. While in the syllabic graph search, significantly more choices than n have to be obtained in order to get n best choices (the likelihoods in the first pass are rough likelihoods). Even though the search is fast in the first-pass, the second pass computing is significantly greater due to an average of 300 choices to be rescored in order to get the correct hypothesis as one of the choices. For this reason, the $A^*$-heuristic lexical search turns out to be faster since there is no rescoring involved (eliminating rescoring overhead) and fewer hypotheses have to be produced.

The disadvantage of this type of search algorithms is that they require large amount of storage. For example, in Kenny's $A^*$ search algorithm, the heuristic scores have to be precomputed. Sufficient storage has to be allocated for the stack to store all possible active hypotheses at any given time. Since the total size of the stack cannot be predicted, the amount of memory necessary can only be estimated by trial and error. In any practical application, the bottom entries in the stack have to be pruned. Enough memory has to be allocated so that truncating the bottom hypotheses does not lead to suboptimal results.

## 2.3 Search Algorithms based on fastmatch algorithm

Another class of search algorithms called the fastmatch search algorithms rely on a fast method to evaluate all the words in the vocabulary in order to generate a short list of candidates. We then use detailed match to evaluate these candidates. The idea here is to get a short list as inexpensively as possible, and to evaluate this short list in detail. For example, in IBM's fast match algorithm [3], they collapse all the states in a hidden Markov model, reducing the model to a one state model. Just this modification reduces the computing by an order of magnitude. For example, for an HMM with 15 transitions, such a simplification reduces number of comparisons from 15 to one. They also use a beam search together with fast match in order to reduce the computing by as much as a factor of hundred. For the beam search, a threshold is used to truncate all the nodes which are unlikely to yield a valid hypothesis. This threshold is manipulated to allow paths of longer length to grow slowly. Such combination of simpler models and beam search reduces the computing by a factor of 100. Obviously, such reduced models require multiple choices to be produced in order to avoid decoding errors. In IBM's 20,000-word isolated-word speech recognizer, a maximum of 100 choices is sufficient in order to keep the percentage of decoding errors to a minimum.

## 2.4 Comparison of various Search Algorithms

The choice of the search algorithm depends on the requirements of the application and the hardware on which the application has to run. For example, Viterbi decoding using a beam search algorithm carries out all the computing frame synchronously, and provides the top choice result as soon as the input is finished. If the primary concern is the real-time delay in recognition, then this is probably the ideal algorithm to use. In systems with large memory, for example workstations, the computing complexity can be managed by tuning the beam search parameters to minimize computing while keeping the loss in recognition accuracy to a minimum. The drawback is that Viterbi decoding requires significant amount of computing and provides only one recognition choice. Post-processing to use more detailed models, segmental features, and language models is not possible with only one choice. These algorithms are generally embedded during the search algorithm, which makes the search algorithm even more expensive. Obviously, a tree-trellis search [29] or some other algorithm can be used after Viterbi decoding to provide multiple choices. The tree-trellis search requires even more memory since intermediate results during Viterbi decoding have to be saved for the tree-trellis search. Also, it introduces delay, since tree trellis search can only be performed after the complete utterance has been received.

$A^*$-search algorithms result in significant reduction in frame-synchronous computing. For example, Kenny et al [20] estimate the heuristic function using Viterbi decoding on a much smaller graph which they call a two-phone look ahead or a one-phone look ahead graph. Obviously, all the heuristic scores have to stored in order to be used during $A^*$ search. For example, for a two-phone lookahead graph, they store the likelihoods for all possible two-phone sequences in the vocabulary for all the frames in the acoustic input. The $A^*$ search also introduces decoding delays, since the search can only start after all the input is in. The delay depends on how rapidly the search proceeds, and this depends on the heuristic function. Even though the average delay in getting top few choices can be small, the maximum possible delay is unbounded. To avoid this maximum possible delay, either the search has to be aborted, or some short cuts have to be taken, resulting in sub-optimal decoding. Most of these parameters can be tuned to limit the loss in recognition accuracy. Because of the large memory requirements, these algorithms are well suited to hardware with large memory, for example workstations.

The fastmatch algorithm actually uses simpler models, in effect reducing memory requirements during frame synchronous search. The simpler models reduce the frame synchronous computing significantly. The beam search during fastmatch does bring some uncertainty to the amount of computing, but this can be controlled by adjusting tuning parameters to allow the longest paths to grow slowly. The fastmatch algorithm does introduce decoding delays, since all the hypotheses have to be rescored using detailed models, segment based features and language model etc. However, this delay can be controlled by limiting the number of choices at the expense of some loss in performance. All these facts show that fastmatch is ideally suited for hardware with limited memory resources.

## 3 DETAILED MATCHING ALGORITHMS

Once we have a short list of word hypotheses, we would like to rescore these hypotheses using best possible algorithms. Since only a few choices have to be rescored, we can use more complex algorithms to rescore them, without incurring heavy computational penalty. Let us look at some of the possible rescoring algorithms.

If the search algorithm used simpler models, then one obvious choice is to use the original models in their full complexity. Other possibilities include using segment based features. For example, imposing constraints based on segment durations and the energy profile of the segments. Where the spoken words are from sentences or phrases, a language model can be used to enhance recognition accuracy. Where a language model cannot be used, some a priori information about the word frequencies, etc. can possibly be used during rescoring.

Many researchers have used segment duration constraints to improve recognition accuracy. For example, Bush and Kopec [8] apply minimum duration constraints on digit-segments to improve digit-string recognition accuracy. They find that a minimum duration constraint of 50 ms for each acoustic segment is optimal for their digit-string recognizer. Soong [28] has used minimum durations to improve phoneme recognition accuracy in spoken Japanese text. He assigns minimum allophone durations to 2084 allophonic HMM's to improve phoneme recognition accuracy.

Many acoustic misrecognitions in the 86,000-word speaker-trained isolated word recognizer at INRS-Télécommunications [11] are due to phonemic hidden Markov models mapping to short segments of speech. When we force these models to map to longer segments corresponding to the observed minimum durations for the phonemes, then the likelihood of the incorrect phoneme sequences drops dramatically. This drop in the likelihood of the incorrect words results in significant reduction in the acoustic recognition error rate. Even in cases where acoustic recognition performance is unchanged, the likelihood of the correct word choice improves relative to the incorrect word choices, resulting in significant reduction in recognition error rate with the language model. On nine speakers, the error rate for acoustic recognition reduces from 18.6% to 17.3%, while the error rate with the language model reduces from 9.2% to 7.2%.

The minimum duration constraints we have found effective depend on the phoneme. For the sonorants and affricates, one duration minimum per phoneme is found to be effective everywhere, while for the remaining phonemes, the duration minimum depends on whether the phoneme occurs in initial, medial or final position in the word. These duration minima vary from 20 ms to 100 ms depending on the phoneme and the context in which it appears. Imposing duration constraints does increase computing requirements. An algorithm to impose these duration constraints efficiently is given in Gupta et al [11].

Segmental energy constraints have also been applied effectively to improve recognition accuracy. For example, Bush and Kopec [8], and Kopec and Bush [21] have applied energy constraints to improve isolated digit and digit-string recognition accuracy. The energy constraints have been imposed on either the peak energy in the speech segment or on the minimum energy in the speech segment. For example, imposing the constraint that the peak energy in a stressed vowel segment be above a certain threshold reduces certain digit insertion errors [8]. Forcing the minimum energy in voiceless fricative segments to be below a certain threshold reduces voiceless fricative confusion with sonorants [21]. These constraints have been incorporated in their recognition algorithm in order to improve recognition accuracy of their isolated digit and digit-string recognizers.

In the 86,000-word isolated-word recognizer at INRS-Télécommunications, we have succesfully applied energy constraints during training of the hidden Markov models for phonemes [11]. Many segmentation errors in the training data are between high energy and low energy phonemes. A number of these segmentation problems have been corrected by constraining the energy contours to prevent phonemes with high energy from mapping onto phonemes with lower energy. Segment boundaries between stops and sonorants, between fricatives and sonorants, between affricates and sonorants, and between breath and sonorants are most amenable to correction using energy constraints [11]. Segment boundaries between vowels, liquids, glides, and nasals can be corrected by using duration minima. Correction of the segment boundaries between phonemes in the training set leads to improved phoneme models, resulting in higher acoustic recognition accuracy and higher recognition accuracy with the language model.

Other successful strategies to improve recognition accuracy include combining results from two independent recognizers. For example, in a flexible vocabulary recognizer for common stocks listed in New York Stock Exchange [24], two separate feature parameters are used. The top choices using cepstral coefficients are rescored using line spectrum pair parameters. The log likelihoods from the two recognizers are then added. This method reduced the error rate from 5% to 4%. IBM has used a similar approach where they use two separate recognizers using different dictionaries. One recognizer uses phonemes to represents words in the dictionary, while another uses fenones to represent the words in the dictionary. Combining the likelihoods from the two recognizers results in a drop in word error rate from 2.5% (for fenonic models) to 1.8% for the combined fenonic and phonetic models [7].

## 4 LANGUAGE MODELS

For each word of spoken input, the acoustic recognizer generates a list of word hypotheses and their associated acoustic likelihoods. The language component takes this probabilistic word lattice as input and uses a statistical model of the syntactic, semantic, and pragmatic properties of English to generate the *a posteriori* most likely word string. A number of language models have been used previously for speech recognition. The trigram language model [17] has been used successfully for both a 5,000-word and a 20,000-word office correspondence task [1]. Derouault and Merialdo [9] use a tri-POS (parts-of-speech) model for a 250,000-word French recognizer. Application of a trigram language model to a recognition task with such a large vocabulary was considered infeasible by Derouault and Merialdo [9]. They also apply global syntactic constraints using a sentence parser. However, application of global syntactic constraints results in only a marginal improvement in the recognition accuracy of their system. The advantage of a tri-POS language model is that it requires significantly less memory for storage than a trigram language model and can be trained from a small training text corpus. However, the tri-POS language model estimates the probability of a word conditioned on the parts-of-speech of the previous two words, resulting in much weaker linguistic constraints than the trigram language model. Lee [23] and Rohlicek et al. [27] have used a simpler bigram or word-pair language model in a 997-word resource management task. To make best use of the available syntactic and semantic constraints, we have used a trigram language model in the 86,000-word vocabulary recognition system at INRS-Télécommunications.

The trigram language model parameters correspond to probabilities of words conditioned on the previous two words. The number of parameters to be estimated is enormous. For example, for the 86,000-word vocabulary recognition system, $86,000^3$ parameters have to be estimated. Even a training set consisting of 60 million words is too small to estimate these parameters reliably. Parameter estimates using relative frequencies would assign a value of zero to a large fraction of the parameters. A number of algorithms exist for estimating parameters for the trigram language model from sparse data. These algorithms include the deleted interpolation method [2], and the backoff method [19]. In the deleted interpolation method, the probability of word $w_3$ conditioned on the previous two words $w_1$ and $w_2$, $P(w_3|w_1w_2)$, is computed as a weighted average of the relative frequencies $f(w_3|w_1w_2)$, $f(w_3|w_2)$ and $f(w_3)$ in the training text corpus. (The relative frequency of $w_3$ conditioned on the context $w_1w_2$ is $f(w_3|w_1w_2) = \frac{C[w_1w_2w_3]}{C[w_1w_2]}$, where the function $C$ counts the number of occurrences of its argument in the text.) The deleted interpolation method requires large amounts of storage for the parameters since both the weights and the relative frequencies are stored. The weights are estimated using the forward-backward algorithm which requires significant computing. The backoff method [19] is storage efficient as it does not require storage of weights. It uses Turing's estimate [10] to compute the probability mass of all the trigrams which do not occur in the training text corpus. This probability mass is then distributed among the unseen trigrams using the bigram and monogram counts. In the implementation of the trigram language model at INRS Télécommunications [14], we estimate the probabilities $P(w_3|w_1w_2)$ for the trigrams $w_1w_2w_3$ which do not occur in the training text corpus by direct application of Turing's estimate, without resorting to bigram and monogram counts to partition the probability mass.

We have found that the trigram language model is the most effective in improving recognition accuracy. Let us discuss some of the issues involved by analyzing our 86,000-word isolated word recognizer. The input to the acoustic recognizer consists of words separated by pauses of at least 150 ms. The spoken text consists of sentences read (without punctuation) from text which was selected randomly from magazines, books, and newspaper articles. An endpoint detector segments the acoustic data $A$ for the spoken word string $W = w_1^n = w_1, w_2, \ldots, w_n$ into subsegments $A_1^n = A_1, \ldots, A_n$ corresponding to the words $w_1, w_2, \ldots, w_n$ in the word-string. For each segment $A_i$, the acoustic recognizer generates a list $\omega_{i1}, \ldots, \omega_{iN_i}$ of $N_i$ most likely word choices, together with their likelihoods $(P(A_i|\omega_{ij}), j = 1, \ldots N_i)$. These likelihoods are smoothed by taking their seventh root before being passed to the second step of recognition. Such a normalization achieves a balance between likelihoods derived from the language model and the acoustic recognizer [4]. The probabilities

$$P(A|W_k) = P(A_1|\omega_{1k_1})P(A_2|\omega_{2k_2}) \ldots P(A_n|\omega_{nk_n}),$$

(where $\omega_{ik_i}$ corresponds to one of the hypothesized words for the acoustic segment $A_i$) are used during search using the trigram language model. This search applies the trigram language model to find the most likely word string $\widehat{W}$ using

$$\widehat{W} = \operatorname*{argmax}_{W_k} P(W_k)P(A|W_k).$$

The acoustic recognizer generates the probabilities $P(A|W_k)$, while the language model provides the probabilities $P(W_k)$.

It is interesting to examine where the language model is most effective in correcting acoustic recognition errors. We analyzed the language model's ability to correct acoustic recognition errors as a function of the coverage of the word and its context by the training set of the language model. Each word in the test set is classified into one of six possible context coverage categories based on the neighboring words found in the training text corpus. To clarify these coverage categories, let us consider the word *sank* in the partial test sentence *every chair sank several inches*. The word *sank* is classified into one of the six categories as follows:

- **five-word context**: if $C[\textit{every chair sank}] > 0$ and $C[\textit{sank several inches}] > 0$ in the training text, then we consider word *sank* to have a five-word context. Note that, we are not implying that the sequence *every chair sank several inches* occurs in the training text nor that the sequence *chair sank several* occurs in the training text. We are only saying that both the sequences *every chair sank* and *sank several inches* are observed in the training text.
- **four-word context**: if $C[\textit{every chair sank}] > 0$ and $C[\textit{sank several}] > 0$, or $C[\textit{chair sank}] > 0$ and $C[\textit{sank several inches}] > 0$ in the training text, then the word *sank* has a four-word context.
- **three-word context**: if $C[\textit{every chair sank}] > 0$ and $C[\textit{sank several}] = 0$, or $C[\textit{chair sank}] = 0$ and $C[\textit{sank several inches}] > 0$, or $C[\textit{chair sank}] > 0$ and $C[\textit{sank several}] > 0$ in the training text, then *sank* has a three-word context.
- **two-word context**: if $C[\textit{chair sank}] > 0$ or $C[\textit{sank several}] > 0$ (but not both) in the training text, then *sank* has a two-word context.
- **one-word context**: if $C[\textit{chair sank}] = 0$ and $C[\textit{sank several}] = 0$, but $C[\textit{sank}] > 0$ in the training text, then *sank* has a one-word context.
- **zero-word context**: if $C[\textit{sank}] = 0$ in the training text, then *sank* has a zero-word context.

In the following text, when we refer to a test word having three-word context, we mean that for this word the conditions outlined are satisfied for the three-word context above, but not for any higher contexts (four-word or five-word contexts). In Table I, we have tabulated how effective the language model is in correcting acoustic recognition errors depending on these contexts. In compiling this table, we only consider words where the word hypothesis list includes the correct word, since only these words can be corrected by the language model. The search algorithm commits 3.4% search errors, therefore, the acoustic and language model recognition accuracies in Table I have been compiled from 96.6% of the test words. The effectiveness of the language model is found to be directly related to the context coverage observed for that word. For example, over 93% of the acoustic recognition errors are corrected for words having five-word context. Most of the remaining errors for these words are due to very low acoustic recognition likelihoods.

The language model is able to correct acoustic recognition errors even for words for which only two-word context is found. That is, the language model is

| Context | Number of words | Acoustic recognition (%) | Recognition with language model(%) | Percentage of acoustic errors corrected |
|---|---|---|---|---|
| five-word | 2000 | 88.3% | 99.3% | 93.6% |
| four-word | 2156 | 85.3% | 97.9% | 85.8% |
| three-word | 1646 | 82.0% | 93.1% | 61.6% |
| two-word | 840 | 81.2% | 85.1% | 20.9% |
| one-word | 306 | 80.4% | 71.6% | -45.0% |
| zero-word | 47 | 87.2% | 59.6% | -216.0% |
| search errors | 245 | | | |
| average | | 84.7% | 94.2% | 62.3% |

**Table 1** Effectiveness of the trigram language model to correct words in the test set depending on contexts occurring in the training text corpus.

effective even when only one of the bigrams (with the word on the left or right of it) has a nonzero count in the training text. One example of a word from our test set having two-word context is *this* in the sequence *have weighed this thought with.* The 60-million word training text has $C[\textit{weighed this}] = 0$, $C[\textit{this thought with}] = 0$, and $C[\textit{this thought}] > 0$. Another example is the word *wives* in the sequence *men without wives whereas the.* In this case, the training set has $C[\textit{without wives}] = 0$, $C[\textit{wives whereas the}] = 0$, but $C[\textit{wives whereas}] > 0$.

The language model increases the recognition errors for words with neither the left nor the right context in the training text. In such cases, the language model does not have enough contextual information. Luckily, only 5% of the words fall in this category. Some examples of such words are *energy* and *wasters* in the sequence *most profligate energy wasters on.* Neither the bigram *profligate energy* nor *energy wasters* occurs in the training text. The word *energy* does occur in the training text. The word *wasters* does not occur even once in the 60-million word training text. Some other examples of words in the test set which do not occur in the training text are *barnburner*, *gondolier*, *heuristics*, *amoebas*, *sashaying*, *luminol*, *Marley*, *doorless*, etc.

## REFERENCES

[1] A. Averbuch, L. Bahl, R. Bakis, P. Brown, G. Daggett, S. Das, K. Davies, S. De Gennaro, P. de Souza, E. Epstein, D. Fraleigh, F. Jelinek, B. Lewis, R. Mercer, J. Moorhead, A. Nádas, D. Nahamoo, M. Picheny, G. Shichman, P. Spinelli, D. Van Compernolle, and H. Wilkens. "Experiments with the Tangora 20,000 word speech recognizer," Proceedings of the 1987 IEEE International Conference on Acoustics, Speech, and Signal Processing, pp. 701–704.

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

[3] L.R. Bahl, S.V. De Gennaro, P.S. Gopalakrishnan, and R.L. Mercer. "A Fast Approximate Acoustic Match for Large Vocabulary Speech Recognition," *IEEE Transactions on Speech and Audio Processing*, vol. 1, pp. 59–67, January 1993.

[4] L.R. Bahl, R. Bakis, F. Jelinek, and R.L. Mercer. "Language-model/acoustic-channel-model balance mechanism," *IBM Technical Disclosure Bulletin* **23**(7B), pp. 3464–3465, 1980.

[5] L.R. Bahl, P.F. Brown, P. de Souza, R.L. Mercer, M.A. Picheny. "Acoustic Markov Models used in the Tangora Speech Recognition System," Proceedings of the 1988 IEEE International Conference on Acoustics, Speech, and Signal Processing, pp. 497–500.

[6] L.R. Bahl, P. de Souza, P.S. Gopalakrishnan, D. Nahamoo, M.A. Picheny. "Decision Trees for Phonological Rules in Continuous Speech," Proceedings of the 1991 IEEE International Conference on Acoustics, Speech, and Signal Processing, pp. 185–188.

[7] L.R. Bahl, P.F. Brown, P. de Souza, R.L. Mercer, M.A. Picheny. "A Method for the Construction of Acoustic Markov Models for Words," IBM Research Report RC 13099 (# 58580), IBM Thomas J. Watson Research Center, Distribution Services 73-F11, 1987, pp. 1–13.

[8] M.A. Bush, and G.E. Kopec. "Network-Based Connected Digit Recognition," IEEE Transactions on Acoustics, Speech, and Signal Processing, October 1987, pp. 1401–1413.

[9] A.-M. Derouault, and B. Merialdo. (1986). "Natural language modeling for phoneme-to-text transcription," *IEEE Transactions on Pattern Analysis and Machine Intelligence* **PAMI-8**(6), pp. 742–749, 1986.

[10] I.J. Good. "The population frequencies of species and the estimation of population parameters," *Biometrika* **40**(3 and 4), pp. 237–264, 1953.

[11] V. Gupta, M. Lennig, P. Mermelstein, P. Kenny, P.F. Seitz, and D. O'Shaughnessy. " Use of minimum duration and energy contour for phonemes to improve large vocabulary isolated-word recognition," *Computer, Speech and Language*, December 1992, pp. 345–359.

[12] V. Gupta, M. Lennig, and P. Mermelstein, "Fast search strategy in a large vocabulary word recognizer," *Journal of the Acoustical Society of America* **84**(6), 2007–2017, 1988.

[13] V. Gupta, M. Lennig, and P. Mermelstein. "Integration of acoustic information in a large vocabulary word recognizer," in Proceedings of the 1987 IEEE International Conference on Acoustics, Speech and Signal Processing, Dallas, pp. 697–700.

[14] V. Gupta, M. Lennig, and P. Mermelstein. " A language model for very large-vocabulary speech recognition," *Computer, Speech and Language*, December 1992, pp. 331–344.

[15] H.-W. Hon. *Vocabulary-Independent Speech Recognition: The VOCIND System.* Ph.D. Thesis, Computer Science Department, Carnegie Mellon University, March 1992.

[16] F. Jelinek. "Continuous speech recognition by statistical methods," *Proceedings of the IEEE* **64**(4), pp. 532–556, 1976.

[17] F. Jelinek. "The Development of an Experimental Discrete Dictation Recognizer," *Proceedings of the IEEE* **73**(11), pp. 1616–1624, 1985.

[18] B.H. Juang, S.E. Levinson, and M.M. Sondhi, "Maximum likelihood estimation for multivariate mixture observations of Markov chains," *IEEE Transactions on Information Theory*, **IT-32**(2), pp. 307–309, March 1986.

[19] S.M. Katz. "Estimation of probabilities from sparse data for the language model component of a speech recognizer," *IEEE Transactions on Acoustics, Speech, and Signal Processing* **ASSP-35**(3), pp. 400–401, 1987.

[20] P. Kenny, R. Hollan, V.N. Gupta, M. Lennig, P. Mermelstein, and D. O'Shaughnessy. "$A^*$-Admissible Heuristics for Rapid Lexical Access," *IEEE Transactions on Speech and audio Processing* **1**(1), pp. 49–58, 1993.

[21] G.E. Kopec, and M.A. Bush. "Network-Based Isolated Digit Recognition Using Vector Quantization," IEEE Transactions on Acoustics, Speech, and Signal Processing, 1985, pp. 850–867.

[22] C.-H. Lee, and L.R. Rabiner. "A Frame-Synchronous Network Search Algorithm for Connected Word Recognition," *IEEE Transactions on Acoustics, Speech, and Signal Processing* **ASSP-37**(11), pp. 1649–1658, November 1989.

[23] K.-F. Lee. *Large-vocabulary speaker-independent continuous speech recognition: The SPHINX System.* Ph.D. Thesis, Computer Science Department, Carnegie Mellon University, April 1988.

[24] M. Lennig, D. Sharp, P. Kenny, V. Gupta, and K. Precoda. "Flexible vocabulary recognition of speech," in Proceedings of the International Conference on Spoken Language Processing, Banff, October 1992, pp. 93–96.

[25] H. Ney, D. Mergel, A. Noll, and A. Paeseler. "Data Driven Search Organization for Continuous Speech Recognition," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, **ASSP-40**, pp. 272–281, Feb. 1992.

[26] N.J. Nilsson. *Principles of Artificial Intelligence*, Tioga Publishing Co., Palo Alto, CA, 1980.

[27] J.R. Rohlicek, Y.-L. Chow, S. Roucos. "Statistical Language modeling using a small corpus from an application domain," in Proceedings of the 1988 IEEE International Conference on Acoustics, Speech and Signal Processing, pp. 267–270.

[28] F.K. Soong. "A phonetically labeled acoustic segment (PLAS) approach to speech analysis-synthesis" in Proceedings of the 1989 IEEE International Conference on Acoustics, Speech and Signal Processing, Glasgow, pp. 584–587.

[29] F.K. Soong, and E. Huang. "A Tree-Trellis based Fast Search for finding the N Best Sentence Hypotheses in Continuous Speech Recognition," in Proceedings of the DARPA Speech and Natural Language Workshop, Hidden Valley, June 1990.

[30] T.K. Vintsyuk. "Element-wise recognition of continuous speech composed of words from a specified dictionary," *Kibernetika*, vol. 7, pp. 133–143, March–April 1971.

[31] A.J. Viterbi. "Error bounds for convolutional codes and an asymptotically optimal decoding algorithm," *IEEE Transactions on Information Theory* **IT-13**, pp. 260–269, April 1967.