NETWORK AND N-GRAM DECODING IN SPEECH RECOGNITION

By

Jie Zhao

A Thesis
Submitted to the Faculty of
Mississippi State University
in Partial Fulfillment of the Requirements
for the Degree of Master of Science
in Electrical Engineering
in the Department of Electrical and Computer Engineering

Mississippi State, Mississippi

October 2000

NETWORK AND N-GRAM DECODING IN SPEECH RECOGNITION

By

Jie Zhao

Approved:

Joseph Picone Professor of Electrical and Computer Engineering (Director of Thesis)	Nicholas Younan Professor of Electrical and Computer Engineering (Committee Member)
James E. Fowler Assistant Professor of Electrical and Computer Engineering (Committee Member)	James C. Harden Graduate Coordinator of Computer Engineering in the Department of Electrical and Computer Engineering
G. Marshall Molen Department Head of the Department of Electrical and Computer Engineering	A. Wayne Bennett Dean of the College of Engineering
Richard D. Koshel Dean of the Graduate School	

Name: Jie Zhao

Date of Degree: October 13, 2000

Institution: Mississippi State University

Major Field: Electrical Engineering

Major Professor: Dr. Joseph Picone

Title of Study: NETWORK AND N-GRAM DECODING IN SPEECH RECOGNITION

Pages in Study: 44

Candidate for Degree of Master of Science

DEDICATION

I would like to dedicate this thesis to my parents and my sister for their support.

ACKNOWLEDGMENTS

I would like to thank Dr. Picone for his guidance through my work and study. I would also like to thank every one in ISIP (Institute for signal and information processing) for their help. A lot of work presented in this thesis is the result of team works. I am only one member of the team. The other team members who have great contributions to the ISIP Speech To Text systems are: N. Deshmukh, A. Ganapathiraju, J. Hamaker, R. Sundaram and Dr. Picone. Thank you all!

TABLE OF CONTENTS

TABLE OF CONTENTS

LIST OF TABLES

TABLE Page

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Automatic speech recognition by machine has been a goal of speech researchers for more than 40 years. In recent years we have seen great advances in speech recognition technology. Some speech recognition techniques have entered into the market place and been used in applications such as command-and-control, credit-card number recognition etc. However, the performance of automatic speech recognition in conversational speech is still far behind human performance [1].

What makes conversational speech recognition so difficult? How does a machine recognize speech? The core of a speech recognition system is the decoder, which finds the most likely word sequence given all the knowledge resources. Decoding is a resource-intensive and time-consuming process. In this thesis, I am going to investigate the efficient implementations for two kinds of decoding processes — the N-gram decoding and the network decoding. At first, let's start with the principles of speech recognition.

1.1. Speech Signal, Phones and Features

A speech signal is a variable of time, amplitude and frequency. An example of speech signals is shown in Figure 1. The top panel shows the waveform of the speech signal, i.e the plot of amplitude vs. time. The bottom one is the wideband spectrogram of the speech. Spectrogram is the three dimensional representation of the speech intensity, in different frequency bands, over time. The wideband spectrogram corresponds to

performing a spectral analysis on 15-msec window data using a broad analysis filter (125 Hz bandwidth) with the analysis advancing in intervals of 1msec. The spectral intensity at each point in time is indicated by the darkness of the plot at a particular analysis frequency. If the spectral analysis is performed on 50-msec window data using a narrow analysis filter (40Hz bandwidth), the spectrogram is called the narrow-band spectrogram [2].

The utterance corresponding to the speech signal in this example is “OKAY HI HI UM”. They can be further decomposed into phones. The term phone denotes the actual sound that is produced in speaking. The ideal linguistically distinct sound unit is called phoneme. In American English, there are about 42 phonemes including vowels, diphthongs, semivowels and consonants. The international standard method to represent phonemes is International Phonetic Alphabet (IPA) [3]. To enable computer representation

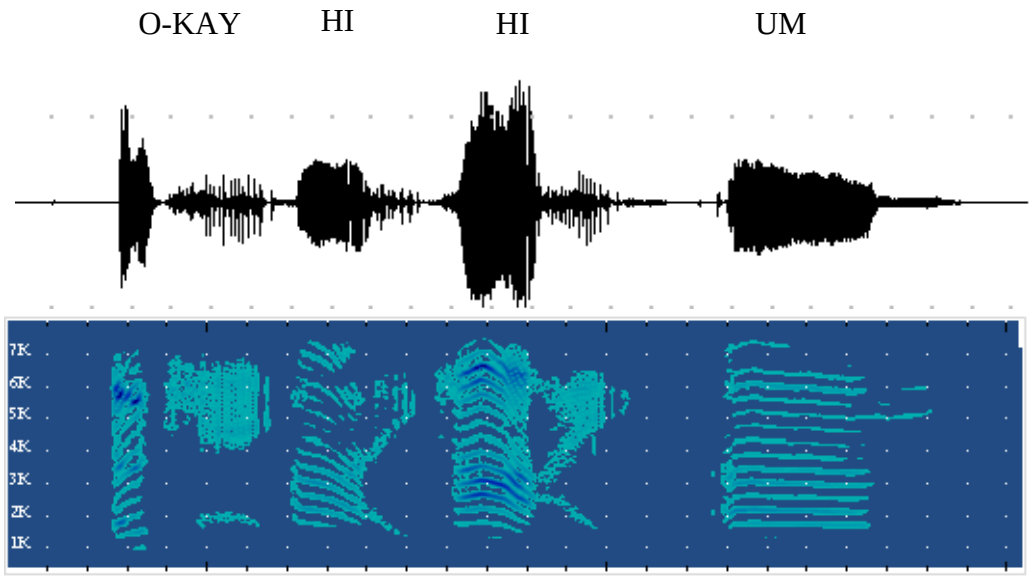


Figure 1. A speech signal and its spectrogram

of the phonemes, it is convenient to code them as ASCII characters. The ARPABET format is usually used. A table shows the IPA and ARPABET representations is included in APPENDEX A [4]. If in ARPABET format, the word sequence “O-KAY HI HI UM” can be represented as “ow k ey h ay h ay um”. Obviously, in Figure 1 the waveforms and the spectrograms of the first phone “ay” and the second “ay” are not the same. Therefore only from the waveform or the spectrogram of the speech, it is impossible to recognize what phones and words are being spoken.

There is another problem for automatic speech recognition. An analog speech signal cannot be input directly into a machine for recognition. It has to be sampled and converted into a digital signal first. After sampling, certain acoustic features are extracted from the sampled speech data. This is done by the acoustic front-end of a speech recognition system. These features are then input into the search engine of the speech recognition system — the decoder, for decoding. These acoustic features are the actual acoustic data the decoder is seeing. For example, if the phone “iy” is uttered, input to the decoder are a set of feature vectors, each of which has multiple coefficients. Ideally speaking, features of the phone “iy” and features of the phone “aa” should be different. If so, our recognition problem would just be an easy pattern matching problem. However, there exists overlap between features of different phones. For example, we selected the features of the vowels “aa”, “iy” from the features of a large amount of speech in Alphadigit database [5]. Figure 2 shows the plot of the first two coefficients of those features. They are separated into 4 groups by gender and phones. From the figure 2, it can be seen that there is a significant feature overlap between vowels “iy” and “aa”. This

means, our measurements to speech data are ambiguous. For the features inside the overlap region, recognition errors can occur. Although higher dimensional features can be used to minimize the overlap, the feature overlap cannot be totally eliminated. In conversational speech there are great pronunciation variations, speaker variations and noises. All of these make conversational speech recognition very difficult. The task of the speech recognition system is to minimize the recognition error rate.

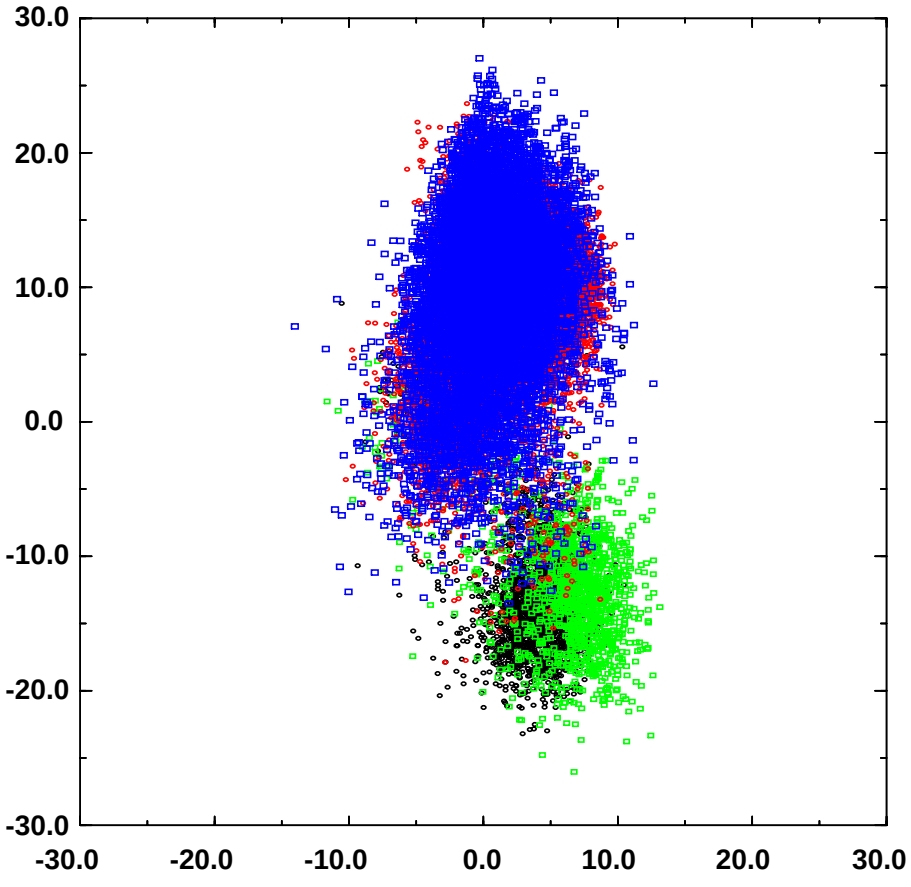


Figure 2. The feature plot of vowels “iy” (blue male, red female), vs. “aa” (green male, black female) from the OGI alphadigit database

1.2. The Statistical Speech Recognition Formula

How to recognize speech given the acoustic data? Most current speech recognition systems use the statistical method [6]. For a sequence of words $W = w_1 w_2 \ldots w_n$, A is the acoustic features which the decoder can use to decide which words were spoken. If $P(W/A)$ denotes the probability that words W were spoken given the acoustic features A was observed, the decoder should choose the word sequences $\hat{W}$ which satisfying:

$$\hat{W} = \arg\max_{W} p(W/A) \tag{1}$$

According to the Bayes rule, $p(W/A) = \frac{P(W)P(A/W)}{P(A)}$, where $p(W)$ is the probability that the word sequence W will be uttered, $p(A/W)$ is the probability that the speaker says W and the acoustic features A is observed. $p(A)$ is the average probability that A is observed. Maximizing (1) is carried out with respect to a fixed A, therefore maximizing (1) is equivalent to find:

$$\hat{W} = \arg\max_{W} p(A/W)p(W) \tag{2}$$

In short, statistical speech recognition is to solve the formula (2) — find the word sequence $\hat{W}$ which maximizes the probability $p(A/W)p(W)$. This formula determines the processes and components of a speech recognition system. First, as mentioned before, a speech signal has to be converted into a format that the decoder can deal with. So an audio front-end is used to extract the acoustic features A from the sampled speech data.

Secondly, the decoder needs to be able to determine the value of $p(A/W)$ that is the probability of producing acoustic features A when the word sequence W is spoken. Hence a statistical acoustic model is needed to model the relationship between a word or its sub-units and the acoustic features. The most popularly used acoustic model is the Hidden Markov Model (HMM), which will be described in the next session. The last part in the formula (2) $p(W)$ represents the language model, which gives the probability of a word or a word sequence are uttered. Based on the above analysis, a basic speech recognition system can be constructed.

1.3. A Speech Recognition System

Figure 3 shows a typical speech recognition system. It consists of three major components: an acoustic front-end, an acoustic model training component and a decoder.

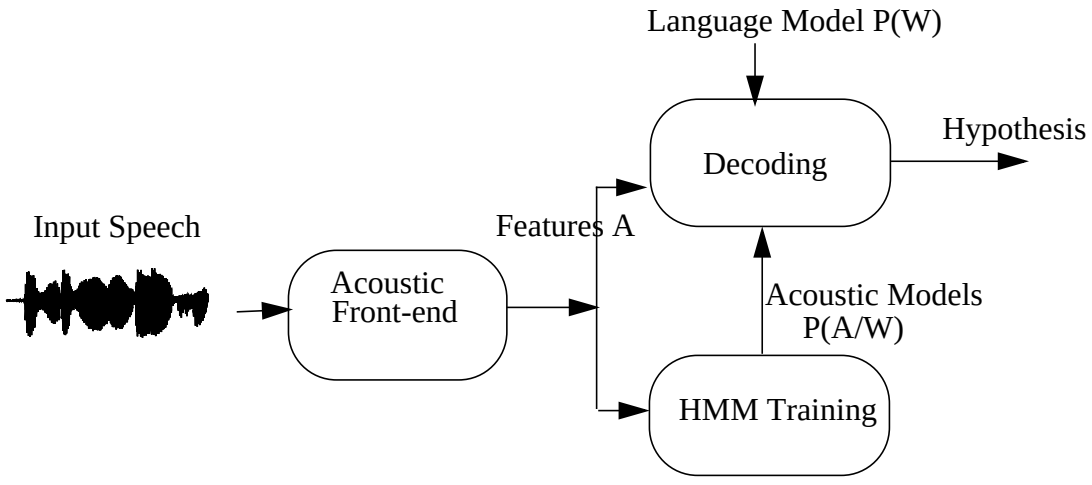


Figure 3. Diagram of a typical speech recognition system

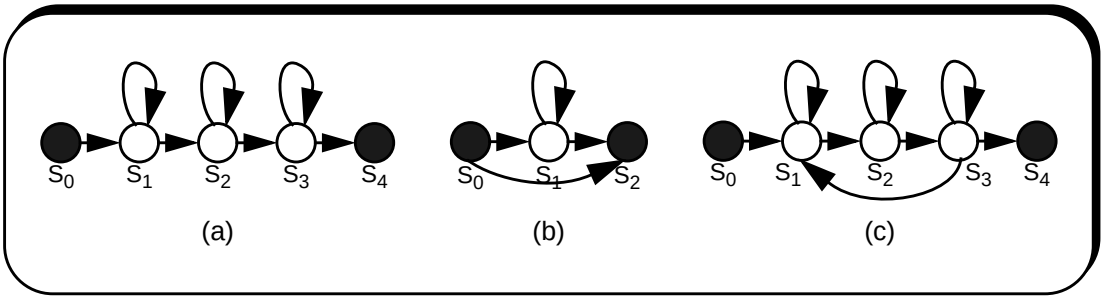


Figure 4. Some typical HMM topologies used for acoustic modeling — a) typical triphone b) short pause c) silence.

The decoder is the core of the speech recognition system. All other components provide the resources for the decoder.

The acoustic front-end provides the acoustic features *A*. It uses signal processing algorithms, such as Fourier transform, window, cepstrum, derivative etc. to extract the features from the sampled speech data [7]. The speech samples are processed in frames of typically 10-15 ms duration and overlapping windows are usually 25-30 ms long. The most popularly used features are the Mel-frequency cepstral coefficients (MFCC) and energy, along with their first and second order temporal derivatives. Typically, each mfcc feature vector is of 39 dimension. Refer [8] for the implementation of the front-end.

The training component provides the acoustic models which hold the knowledge of $p(A/W)$. The standard acoustic model is the Hidden Markov Model (HMM). An HMM corresponds to a sub-word unit such as a phone (context-independent or context-dependent) or a syllable. Each HMM consists of 3 or 5 states where each state models the onset, middle and end of the phonetic context respectively. Figure 4 shows some examples of the various HMM topologies used for acoustic modeling in speech

recognition. As shown in the figure, each state can transit to the next state or loop back to itself. This represents that the length of uttering a phone can vary. There is a statistical model of the weighted mixtures of multivariate Gaussian distributions associated with each state. The input feature vectors are evaluated by these statistical models and produce the probability $p(A/W)$. Originally, the statistical models of the states of HMMs are unknown. This requires training to estimate the parameters of the statistical models, including the mean, variances and the mixture weights of the Gaussian distributions. Accurately training HMMs needs a large amount of speech data from known speech. The commonly used training algorithms are Viterbi algorithm and Baum-Welch forward backward training algorithm [9]. Theoretically speaking, the HMM of a certain phone, say "aa" should give a higher probability when evaluating the features of the phone "aa" than evaluating other phones. Therefore the correct word sequence would have the highest probability and be correctly recognized. However, this is not always true because the acoustic models being trained can not be perfect. In addition, the problems addressed before such as pronunciation variations, feature overlap, noises etc. will also result in recognition errors.

Another knowledge input to the decoder is the language model (LM). The language model constrains which word can follow which word and provides its probability. It integrates linguistic knowledge, domain knowledge and any other pertinent information to constrain the word sequences being recognized. Since the probability of a word being spoken often depends on the words spoken previously, this gives rise to the N-gram LM. In the N-gram LM, each word can follow each other, and the probability of

the current word only depends on the *N-1* previous words. Typically, one-pass decoding using an N-gram language model is called N-gram decoding. N-gram decoding is very resource-intensive and time-consuming. Sometimes an explicit network can serve as an LM. This network can be a grammar that defines the structure of language used in the recognition task, or a word graph generated by a previous recognition pass. The network limits the search space to a relatively small amount of possible word sequences. Only the word sequences in the network can be recognized. Decoding using the explicit network as

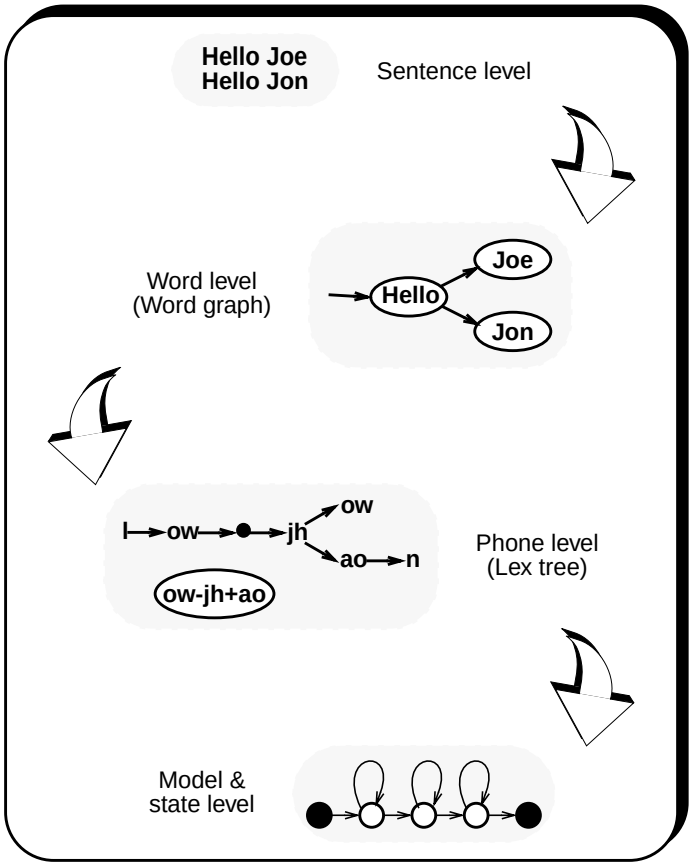


Figure 5. Hierarchical representation of the search space

the high level constraint is called the network decoding. The N-gram decoding and the network decoding are the focuses of this thesis.

After obtaining speech features, acoustic models and language models, we are ready for decoding. Decoding is to search the most possible word sequences given the acoustic features. The search space is a large network composed of a hierarchy of sentences, words, phones and states of HMMs. Figure 5 displays the hierarchical structure of the search space. The number of possible word sequences is quite large even for a small vocabulary task. Therefore for large vocabulary speech recognition (LVCSR), an exhaustive search cannot work and an efficient search paradigm needs to be employed. Here, let me give a brief introduction on the typically-used search algorithm — the Viterbi search algorithm.

1.4. Viterbi Search Algorithm

Viterbi search and its variant forms belong to a class of breadth-first dynamic programming techniques. Here, all hypotheses are pursued in parallel and gradually pruned away as the correct hypothesis emerges with the maximum score. In this case, the recognition system can be treated as a recursive transition network composed of the states of HMMs in which any state can be reached from any other state. The Viterbi search algorithm builds a breadth-first search tree out of this network by the following steps:

1. Generate a list of all states $S(t)$ for each frame t of the utterance.
2. Initialize each list by setting the probability of the initial state to 1 and the rest to 0.

3. For each state $s \in S(t)$,

   for each possible transition from s to some state $s' \in S(t)$

   - If the list for s' is uninitialized, initialize it with the transition score $p(s'/s)$ and a back-pointer to s.
   - Else update the score for s' only if this transition gives a better path score.

4. Repeat step 3 with $t = t + 1$.

   - If $t = N$, the utterance duration, then trace back to get the best path.

Viterbi search is time-synchronous, *i.e.*, at any stage, all partial hypotheses generated during the search terminate at the same point in time. Since these hypotheses correspond to the same portion of the utterance, they can be directly compared with each other. Viterbi search is impractical for even moderate-sized tasks because of the large size of the state space. Therefore, other constraints such as language models and pruning techniques have to be applied to reduce the search space.

1.5. ISIP Speech-to-Text System

All the work in this thesis is based on the Institute for Signal and Information Processing (ISIP) public-domain speech-to-text (STT) systems [10]. I am one of the key members of developing and enhancing them. There are two notations about the ISIP STT systems. The current ISIP STT system, which is publicly available, is referred to as the prototype system. This prototype system was started in 1998, and has been continuously enhanced. Now it has become a complete recognition system, and included the following

modules: an audio front-end, Baum-Welch (BW) training, Hidden Markov Model-based (HMM) state-tying, and a hierarchical decoder based on the Viterbi algorithm. The functions of the prototype decoder include network decoding (or word-graph rescoring), word-graph generation and N-gram decoding etc. I have worked on developing the front-end, the decision tree-based state-tying and the network and N-gram decoding parts of this prototype STT system. The N-gram decoding algorithm is the focus of Chapter 2.

The other ISIP speech recognition system is called the production system. Our goal is to combine our expertise in decoding algorithms and C++ programming language, make the new system highly generalized and flexible, and make it easier to use and enhance. The production STT system is based on our extensible and flexible ISIP Foundation Classes (IFCs). The ISIP Foundation Classes and software environment provide researchers a friendly and stable environment, save their time to rewrite some common functions and data structures. The IFCs consist of system, I/O, math, data structure, algorithm, signal processing, and many more such useful classes. We have been developing IFCs for over one year. They construct the solid ground for our production decoder. The development of the production decoder started from the beginning of this year. As the first step, we added the network decoding capability into the production decoder. The preliminary experiments of the production-format network decoding have shown promising results comparing with the prototype system. The network decoding and some design issues about the production system will be talked in Chapter 3.

1.6. Organization of Thesis

The goal of this work is to investigate the correct and efficient implementation of the network and the N-gram decoding processes. The work is based on the ISIP prototype Speech-To-Text system and the ISIP foundation classes.

CHAPTER 1 gives an introduction to the speech recognition problem. It provides basic knowledge about the components of a speech recognition system and the decoding algorithms.

CHAPTER 2 describes the N-gram decoding based on our prototype decoder. It focuses on how to apply N-gram language models into decoding. Some precious experiences we gained through our work are addressed. CHAPTER 3 talks about the generalized network decoding and its design in our production decoder. This is the first step for us towards building the production STT system. The implementation is based on the ISIP foundation classes (IFCs). In CHAPTER 4, both systems are evaluated.

CHAPTER 5 concludes this thesis and points out some future work.

CHAPTER 2

DECODING USING N-GRAM LANGUAGE MODEL

In chapter 1, we have looked at the typical structure of a speech recognition system. Acoustic models and language models (LM) are two knowledge resources input into the decoder. For large vocabulary continuous speech recognition (LVCSR), the possible word sequences are extremely large. Exhaustive search is impossible. Therefore a language model is required in order to constrain the search space during the decoding process. N-gram language model is the dominantly used language model since it provides a relatively compact representation of the linguistically probable word sequences. An N-gram LM can be applied into several types of decoding. This chapter deals with those types of decoding which use N-gram language models.

2.1. N-gram Language Model

Every language consists of a sequence of words. A language model is to provide the probability of next words given preceding words. An N-gram language model uses the history of the $n-1$ immediately preceding words to compute the occurrence probability P of the current word. The value of N is usually limited to 2 (bigram model) or 3 (trigram model) for feasibility. If the vocabulary size is M words, then to provide complete coverage of all possible word sequences the language model needs to consist of M^N N-grams (*i.e.*, sequences of N words). This is prohibitively expensive (*e.g.*, a bigram language model for a 40,000 words vocabulary will require 1.6×10^9 bigram pairs), and

many such sequences have negligible probabilities. Obviously, it is not possible for an N-gram language model to estimate probabilities for all possible word pairs. Typically an N-gram LM lists only the most frequently occurring word pairs, and uses a backoff mechanism to compute the probability when the desired word pair is not found.

For instance, in a bigram LM, given w_i, the probability of the next word is w_j:

$$\hat{p}(w_j|w_i) = \begin{cases} p(w_j|w_i) & (w_i, w_j) \text{ exists} \\ b(w_i)p(w_j) & \text{otherwise} \end{cases} \quad (3)$$

where $b(w_i)$ is the back-off weight for the word w_i,

$p(w_i)$ is the unigram probability of the w_i

The backoff weight $b(w_i)$ is calculated to ensure that the total probability:

$$\sum_j \hat{p}(w_i, w_j) = 1 \quad (4)$$

Similarly, for a trigram of words $w_h w_i w_j$,

$$\hat{p}(w_j|w_h w_i) = \begin{cases} p(w_j|w_h w_i) & (w_h, w_i, w_j) \text{ exists} \\ b(w_h w_i)\hat{p}(w_j|w_i) & \text{otherwise} \end{cases} \quad (5)$$

N-gram language models have been effective in improving the efficiency in LVCSR significantly. However, even though N-gram language models store only a small subset of all the possible sequences of N words, they are still significantly large for large vocabulary applications. The language model score calculation is quite an expensive process in N-gram decoding. Many systems have been devised which try to improve the efficiency of N-gram decoding [14, 15]. In [16], an entropy-based pruning algorithm has

been applied to prune the language model size while not losing much information. In addition to N-gram LM, statistical language models [17] are also being investigated by some universities. But so far, they are still not as popularly used as the N-gram LM.

2.2. Applying N-gram LM into Decoding

The usage of an N-gram LM in decoding is to provide the probability of one word following the others. How to apply this depends on the type of decoding. According to the number of decoding passes, there are two types of decoding: one-pass decoding and multi-pass decoding. That means, the final best hypothesis may or may not be obtained from a one-pass decoding. An N-gram LM can be used directly as the constraints to the search space. In this case, the search network is a fully connected word network consisting of all the possible combinations of word sequences. The word sequence probabilities are restricted by the N-gram LM. If only the one best hypothesis is generated by searching though such a network, the decoding is called one-pass N-gram decoding. An alternative to the one-pass N-gram decoding is the word-graph generation. In the above process, instead of keeping only the 1-best hypothesis, it keeps multiple hypotheses at each word end, and outputs them as a word graph. This word graph contains a set of nodes and arcs that form a large number of possible word sequences. There are both language model score and acoustic model score associated with each arc. The word graph then can be used in a second-pass decoding as the language model constraints. The second-pass decoding is also often referred to as the word-graph rescoring or network decoding. Since the word graph is much smaller than the fully connected word network, word-graph rescoring is

much more efficient than the one-pass N-gram decoding. Although the process of word-graph generation is time-consuming, once a word graph is generated, it can be rescored using more complex acoustic models and language models. For example, in the first pass, we can use the word-internal triphone HMMs and bigram LM to generate the word graphs. In the second pass, we rescore the word graphs using cross-word triphone HMMs and a trigram LM. This will give an overall better performance in terms of both speed and accuracy. In word-graph rescoring, to obtain the word sequence probabilities, we can either use the original LM score associated with the graph arcs which were obtained during word-graph generation or use N-gram LM again. In the latter case, we only use the graph structure while throw away the language model score associated with graph arcs. The word sequence probabilities are calculated from the N-gram language model.

In short, N-gram language models can be used in one-pass N-gram decoding, word-graph generation or word-graph rescoring. In all the above cases, the common task is how to efficiently apply the LM into decoding. Efficient N-gram decoding is very difficult. It is a trade-off between accuracy and speed. The difficulties involve how to efficiently store and look up the LMs, calculate the LM scores and correctly apply them.

2.3. N-gram Organization

Because of the large amount of N-gram models, the representation of N-gram model data structure has direct effect upon the memory size and the run-time speed. The storage structure should occupy less memory and be easy to look up.

Use a trigram LM (N=3) as an example. The unigrams (1st order gram) in this LM are just single words. The bigrams are word pairs, whose second word follows the word in unigram. So we can group all the words following the same 1st word together, point them to the unigram word, and only store the second word and its corresponding bigram probabilities. For trigrams in this LM, it can be stored similarly. All the third words following the same bigram are grouped together. Only the third word is stored, having a pointer to its preceding bigram. Figure 6 shows a data structure organized as the description above. Each order N-gram is stored as a list. Each list is an array of N-gram nodes. Each unigram node points to the head of a set of bigram successors, and each

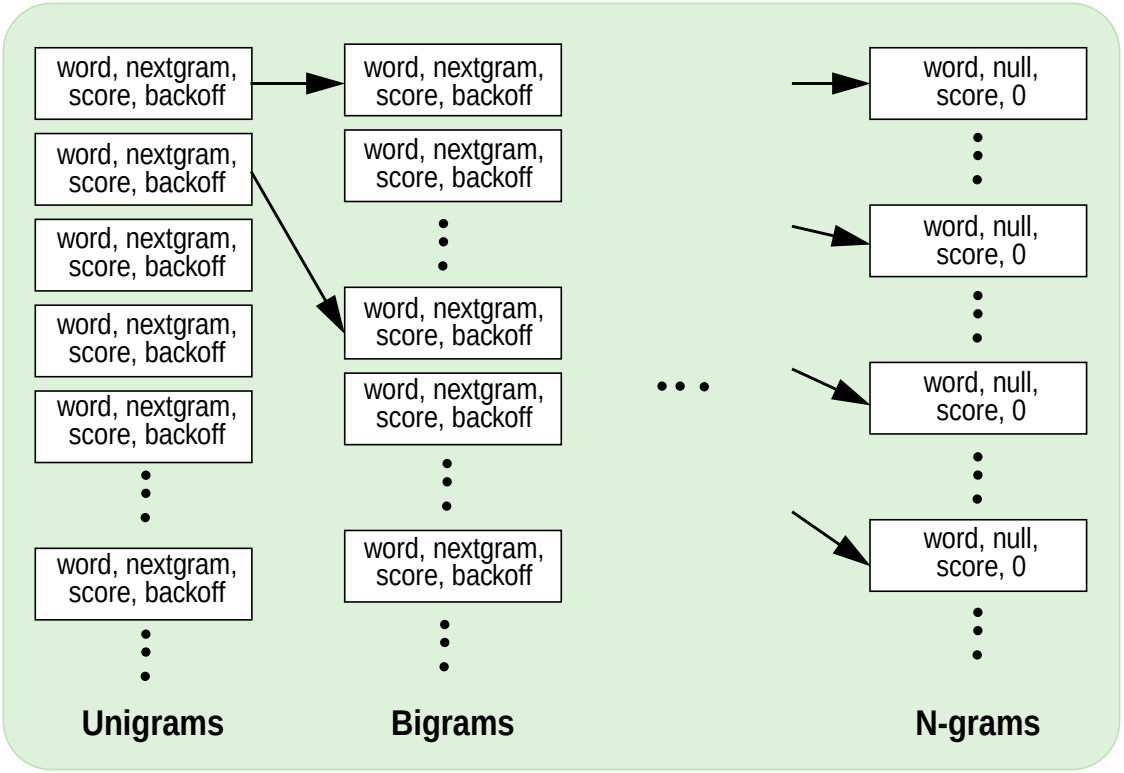


Figure 6. N-gram language model storage structure

bigram node points to the head of its trigram successors. An N-gram node has the following components: word identity, N-gram probability, a backoff weight, and a pointer to the head of the next order N-gram successors. The word here only records the last word of an N-gram. This scheme can save a large amount of space to store the words. For LM score calculation, the first word is found first, then the next. If the word sequence is not present in the explicit LM, backoff scheme is used to calculate the LM score.

LM lookup can be speeded up greatly if an index instead of a string is used to represent the word identity in the above data structure. In this way, when looking up word sequence in the LM, no string comparison is needed. This will significantly improve the LM lookup speed, since integer comparison is much faster than string comparison in C. The storage can be further minimized if the highest-order N-gram list is distinguished with others, because the highest-order grams do not have backoff weights and pointers to their next N-gram list. In addition, a lookup table can be used to minimize the total cost to store the LM and backoff score [18]. That means to quantize those float numbers and limit them to a certain amount.

2.4. N-gram Histories

Understanding N-gram histories is a key to correctly apply N-gram LM into decoding. During decoding, when two paths reach the same point, do we merge them or propagate them further as separate paths? The answer to this question is simple: merge them when they have the same history. If so, only the path with a higher score can survive. However when the search space involves words, phones and states, the path histories

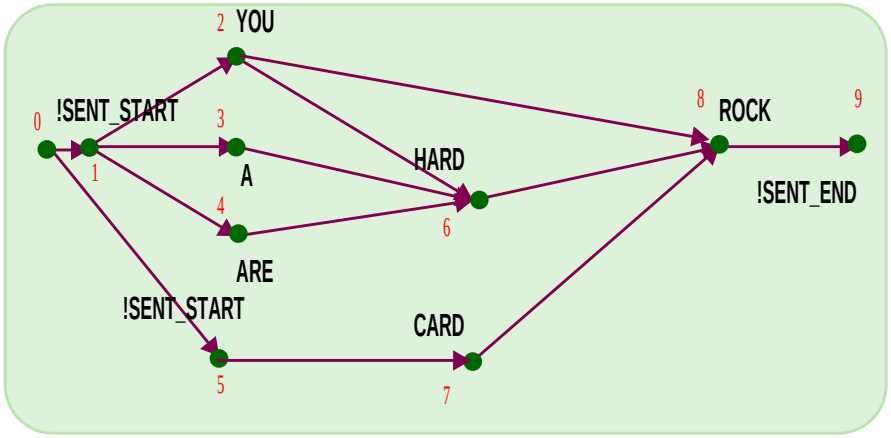


Figure 7. An example of a word graph that demonstrates how the LM is applied when rescoring paths through this graph.

become confusing. In decoding using N-gram language models, at a cross point, if the N-1 previous words (nodes) in the paths are all the same, we regard these two paths have the same history at the word level.

Path merging has a profound impact on the computing resources (particularly memory) required for LVCSR. For decoding using a bigram LM, there is only one word history. It is easy to handle. For trigram and above LM, it needs to pay attention to how to handle the N-gram histories. For example, considering the word-graph rescoring problem shown in Figure 7. The length of the history plays a major role in whether paths are merged, and how efficiently we can implement the merging process. In this word graph, there exist paths “YOU HARD ROCK”, “A HARD ROCK” and “ARE HARD ROCK”. Each reaches the point “ROCK”. The decision to merge them depends on the length of the history and/or the type of LM (bigram, trigram, and network will produce different results). If a bigram LM is used, at the point “ROCK”, only the previous one word needs to be considered. Therefore when these three paths reach “ROCK”, since the previous

word for “ROCK” in these three paths is at the same graph vertex (e.g., “HARD”), only one path can survive and be propagated further. On the other hand, if a trigram LM is used, the previous two words in the path have to be considered. Therefore these three paths have different histories and cannot be merged. An easily made mistake would be not considering enough history words. This was also an error in the system we submitted for the Hub-5E Eval’2000 [12], in which our History object only has a pointer to the previous one word/node. For the bigram case, this happened to be correct. But for the trigram case, paths with different histories, such as “YOU HARD ROCK”, “A HARD ROCK”, were treated as having the same history. This resulted in a search error: extra path merging, over-pruning and hence an incorrect likelihood calculation.

In our Hub-5E Eval system, only a bigram LM was used in the first pass for word-graph generation, so this problem did not affect the resulting word graphs. However, in the second pass, these word graphs were rescored with a trigram LM. This flaw essentially negated the potential benefits of using a trigram LM. This error was corrected by maintaining the previous N-1 words in the History object rather than only the previous word. Depending on the decoding types (N-gram decoding or word-graph rescoring), the previous words may come from a string word, a unigram node, or a word-graph node which has a word entry. One thing to mention is that, even in N-gram decoding mode, one can not just use a pointer to a lower order N-gram node as the history thinking that it can give you the previous N-1 words. For example, if we are doing trigram decoding, why not use a pointer to a bigram as the history? The reason is that not all word pairs have their

corresponding bigram node! That is also why the N-gram history issue is intuitively simple, but needs great attention to handle.

Now in our ISIP prototype STT system, the trigram LM scores are correctly applied. For the system we used for Hub-5E 2000 evaluation, fixing this bug reduced the WER by 0.5%. However run-time memory usage was increased by almost 30%. The drop of WER with this correction was not as significant as we had expected. The reasons are:

- The language mode we used, which was provided by SRI, had been heavily pruned [20]. The OOV (Out Of Vocabulary) percentage is high.
- The word graphs we generated in the first pass have a very high error rate. So in the second pass, even the trigram is correctly applied now, its effect can not be fully accounted. The experiment results will be provided in Chapter 4.

2.5. N-gram Lexical Tree

In LVCSR, the lexicon is often organized as a tree since many words share the same start phone, and search the first phone of each word takes the most effort [21]. The tree-structured lexicon is called lexical tree. It can save a lot of computation and storage.

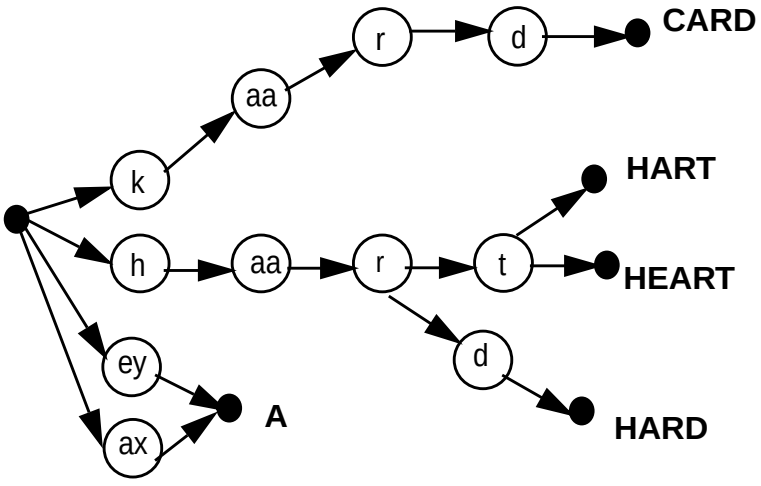


Figure 8. An example of a lexical tree

An example of a lexical tree is shown in Figure 8. A lexical tree is used to generate phone hypotheses. Each lexical node contains a list of the words on that path covered by the monophone held in the lexical node. The dark circles represent starts and ends of words. In order to compute LM scores, each lexical tree also needs to maintain its word histories. The scores for word transitions are computed based on the position in this lexical tree and the current word histories. A problem of lexical tree is that the word identity is unknown until a word-end lexical node is reached. When there is a transition from W_1 to W_2, the word W_2 is unknown until the end of the lexical tree. Therefore the LM score can not be applied immediately upon the transition. That means pruning based on language models can not be applied as early as possible. An LM lookahead [22] technique is devised to overcome this problem. The highest language model score at each lexical node is acquired based on the current word(s) and the history word(s), and used as the LM score at that point.

Typically, in lexical tree-based searches, the LM scores are stored in the lexical tree nodes. At each lexical node, a list of next words is maintained. When a word end is reached during the decoding process, the decoder constructs a new lexical tree encompassing all the possible next words that are used to generate the next phone hypotheses. For word-graph rescoring, constructing new lexical trees is necessary since the possible next words are different for each word, and the number of possible next words is relatively small. However in N-gram decoding and word-graph generation, each word (except the sentence start word) can be followed by any other word. So for large vocabulary speech recognition, the lexical tree would be very large. Even a few copies of

the complete lexical tree will quickly overshoot the available memory. Since we know that the N-gram lexical tree structure is the same for all occurrences of each word, and that only the LM scores vary according to their N-gram histories, making many trees with the same structure is a waste of both time and memory. Therefore, instead of making many copies of the lexical tree, we reuse the same N-gram lexical tree for each occurrence of a word, disconnect the history words and LM scores from the lexical tree.

A data structure called instance is introduced to represent the unique position in the search space which is determined by the corresponding history word(s), the lexical node and the phone (the lexical node). LM lookahead is performed at each lexical node, and the highest language model score is stored in the instance. Two instances can be merged if they have the same word histories, the same lexical node and phone, i.e. at the same position in the search space regardless of time. Now when a word end is reached, this word and its N-2 previous history words (if any) will become the new history for the coming word. Since the instances are reused in the decoder, the score calculation needs to be done only once. This minimizes the total access times to the N-gram LM.

2.6. Pruning

To save the computation and memory, it's imperative to stop those less-likely partial paths from growing further. That is also the reason we use LM lookahead, so we can apply severe pruning earlier. The idea of pruning is to only prune away those unlikely paths. However it's possible that a path with a lower score in the early frame may grow to be the best path later. If the best path is pruned way, it will result in a recognition error.

There are many pruning techniques [23]. We used the followings in our decoder. The first one is the well-known beam pruning technique. A path is pruned away if its path score is below the current best hypothesis score minus a specified beam width. Beam width can be different at each level in the search hierarchy. Usually heavier pruning is applied at word end because there are much more possible next words at word end than possible next phones at phone end. Secondly, we set a maximum number of words allowed at each frame. This is also to avoid fan-out at the word end. Thirdly, instance pruning is applied at each frame, which is limiting the number of model instances active at a given time. The instances are sorted according to their scores. Only the instances having better scores can survive. Pruning has dramatic impact on the search speed and memory cost. The trade-off is accuracy. Heavy pruning with less decreasing in accuracy is important for efficient N-gram decoding and word-graph generation.

In all, N-gram decoding is a complicated and resource intensive process. Although our decoding and pruning approaches improved the decoding efficiency and accuracy, they need to be and can be improved further. Also we realized it is important for a speech recognition system to be extensible and flexible, because it often happens that we found a flaw in our original design, or we wanted to add new modules to the system. For a public-domain speech recognition system, this need is more imperative. Therefore developing a more generalized and flexible system while keeping its efficiency is our next objective. We started to develop our production speech-recognition system in spring 2000. As the first step, we developed the network decoding in a new fashion. The next chapter is going to show how to design such a decoder.

CHAPTER 3

NETWORK DECODING

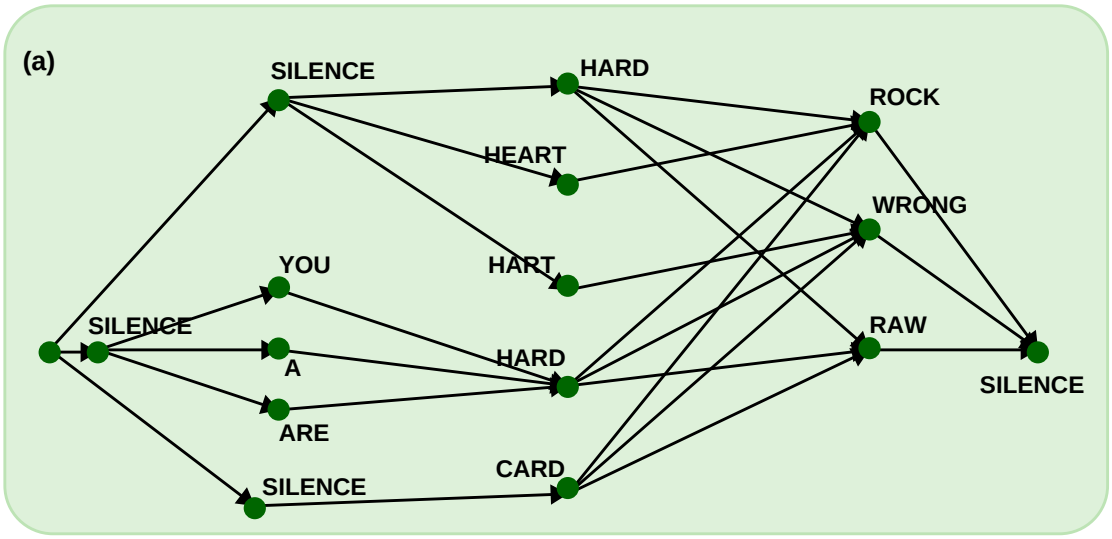
3.1. Network Decoding

The network here means a word graph, i.e. a network of words. It is a subnet of the fully connected word network, as the one in N-gram decoding, and provides constraints on search space at the word level. This network could come from the first-pass word-graph generation as mentioned in the last chapter. If so, this second-pass network decoding is often called word-graph rescoring. The network could also be constructed by some grammars. For example, a network of telephone numbers should follow some rules (grammars) such as area code, local phone number, to make the phone number generated by this network valid.

As mentioned in Chapter 1, each word can be separated into phones. Each phone can be modeled by a Hidden Markov Model (HMM). An HMM consists of several states associated with some statistical distributions. The final search space is actually composed by the expansion of these hierarchical networks. Fig. 9 illustrates the expanded search space up to the phone-level. The term “triphone” denotes a context-dependent phone, which has one left context and one right context. A phone without context is called monophone. A word-internal triphone denotes that the contexts of a triphone are restricted within a word. On the contrary, if its contexts can come from neighbor words, the triphone is called a cross-word triphone. Of course, for each phone, either monophone,

word-internal triphone or cross-word triphone, it needs corresponding HMM to model it. The parameters of HMMs (mean, variance, and/or mixture weights of the statistical models) came from training procedure [24]. The input speech features are evaluated by each state of an HMM. This produces the probability of those states generating such features.

The decoder is to search the best match to the speech features among many paths. It's like there are many path markers travelling from the start point to the end point. A path marker remembers which path it came from, and represents a unique path. The total travel time is fixed (the length of data). While which route a path marker goes though and how long a path marker stays at each state is not fixed. Path markers can split and merge. For example, at time t, a path marker is at state s. At time t+1, the path marker may stay at state s or go to the next state. Therefore the marker splits into two markers, which will have different routes from now on. Their path scores are the accumulated scores obtained by evaluating the speech features at a certain state and a certain time. If two path markers arrive at the same state at the same time, only the one with a better path score can go further. If a path marker arrives at the end of all the states belonging to a word, it will split again and enter into next words as restricted by the word graph. At that point, language model scores will also be accumulated into the path scores. Finally, time ends. Many path markers reach the end. Among those paths, the one with the highest score wins, because along this path it's most likely to produce the speech features matching the input one. The states, phones, and words along the best path construct the final hypothesis. Note that during the search process, there may exist two or more path markers that reach the



(b)

SILENCE	sil	HART	h aa r t
A	ax	HEART	h aa r t
A	ey	RAW	r ao
ARE	aa r	ROCK	r ao k
CARD	k aa r d	WRONG	r ao ng
HARD	h aa r d	YOU	y uw

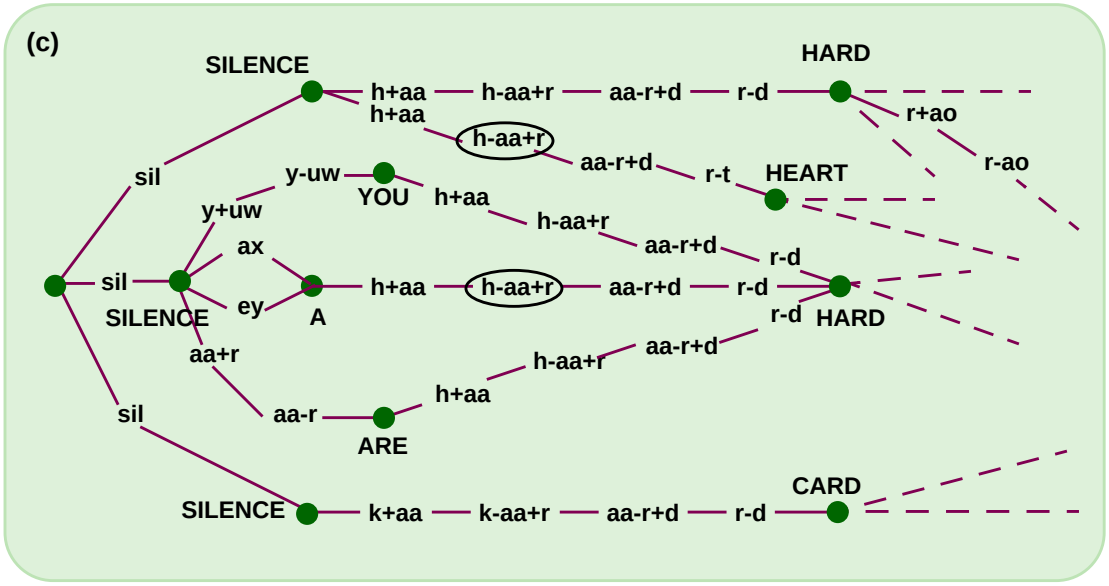


Figure 9. An example of expanded network. (a) the word level network (b) the pronunciations for the words involved (c) The network expanded using the corresponding word-internal triphones

triphone hh-aa+r at the same time. However, they cannot be merged into a single path because they belong to different words in the network at the word level.

The network shown in the above example consists of only three levels: word, phone and state. In fact, the number of hierarchies of a network is not only limited to three. It could also have sentence level, syllable level etc. No matter how many levels it has, no matter what each level is, the search space is always a hierarchical network. Each level has one or more graphs or subgraphs. The search space can be expanded down to the bottom level. Search is hence performed on the expanded network. Therefore the decoding process can be viewed as a general search process over the expanded network. Our production decoder is designed to reflect such a general decoding process. It is based on the ISIP Foundation Classes (IFCs).

3.2. ISIP Foundation Classes

Not many researchers and students can have their chances and time to write their own decoder, because there is a huge overhead before they can get to the point of writing the core decoding algorithms. They have to deal with IO with various files, such as the audio file, lexicon, states file, models file etc. They have to worry about the data structures, the command line parsing etc. Therefore we aim to provide users and ourselves an easy and powerful software environment to work on. This is the ISIP foundation class (IFC).

IFC is a highly object-oriented hierarchical package [25]. It is built from bottom-up. It consists of system, IO, math, data structure, shell etc. libraries. Memory is managed by a center MemoryManager class. An important class in the IO library is Sof

(Signal Object File). Sof is a unanimous file format used across all the objects in IFC. All objects can read and write themselves into Sof file. The function format is consistent across all the classes, hence IO becomes very easy. The data structure library has commonly used data structures such as node, linked list, hash table, graph, etc. Fig. 10 shows the hierarchy of ISIP Foundation classes. Below the algorithm library, the classes are not speech recognition specific. They can be used for general-purpose programming. In the algorithm library, we developed algorithms useful for speech recognition, such as

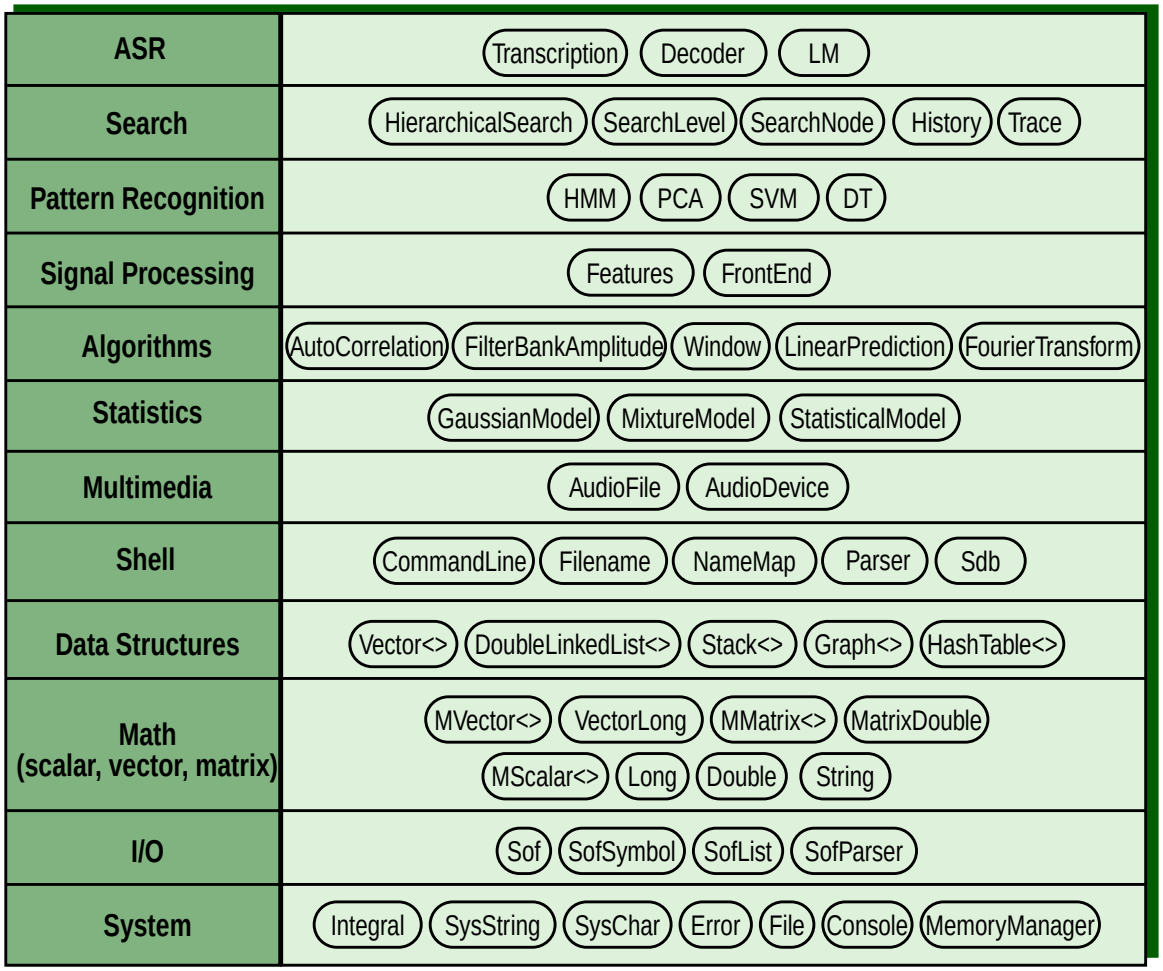


Figure 10. Hierarchy of ISIP Foundation classes

FourierTransform, LinearPrediction and Cepstrum. Users could use lower level IFCs to develop their own algorithms also. For reading / writing audio files in any format, there is the AudioFile class. The statistics library is used for evaluating data based on statistical model. There are many other such useful classes. They are very easy to use and modify. On the top of IFC, it is the search library, which is the core of the production speech recognition system. With all the low-level modules handy, we can focus on the decoding algorithms. With the experiences we gained during developing the prototype system, we are able to incorporate the efficiency and generalization issues into the design of the production decoder very early.

3.3. Production Decoder

The search library is to implement a generalized hierarchical search algorithm. HierarchicalSearch and SearchLevel are the two main classes in the search library. HierarchicalSearch consists of any number of SearchLevels, such as a sentence level, word level, phone level and a state level. Every SearchLevel is essentially the same as the others. All of them can be represented by graphs. At the sentence level, it has a sentence graph. At the word level, it has one or more word graphs, each of which is the subgraph of a sentence in the sentence graph. At the phone level, there are some phone graphs to represent the pronunciations of each word. Finally at the state level, the HMMs are also graphs and the subgraphs of states. Each SearchLevel holds one or more sub-graphs at that level. Connecting these graphs builds the entire search space. We store it as a Graph<SearchNode>, where SearchNode is an embedded object inside the GraphVertex.

Each SearchLevel also holds all the search symbol strings at that level. Thus in the Graph<SearchNode>, each SearchNode can be referenced by the symbol index instead of a string symbol. This avoids string comparisons and makes fast decoding possible. The SearchNode object also has a pointer to a StatisticalModel object. Thus, data evaluation can be done at each SearchNode at any level.

The network decoding has been implemented in the production decoder. Each search graph is expanded dynamically at the run time. A path marker is propagated from the top level down to the bottom level, then propagated up. A path marker splits to several path markers if different paths are taken. The path score is updated during propagation. Viterbi pruning takes place at the lowest level after evaluating each statistical model. Path markers remember their histories. Only path markers with the same history can be compared and merged. The path marker with the best score can be propagated further; others are deleted. After traversing up, pruning can also be applied at each search level.

The decoding algorithm is the same as the one in our prototype system. It's still a hierarchical Viterbi decoding algorithm. We have made the context-independent network decoding working. Its results could fully match the prototype decoder. The code in the production decoder is more modular and user friendly. It is easier to modify and extend. As the next step, we will do more complicated tasks, such as N-gram decoding, word-graph generation. In those cases, fully expanding the hierarchical network is impossible. More efforts are needed to manage the size of the search space. Some concerns are:

- How to represent a lexical tree? In a lexical tree, many pronuciation graphs merge into one graph. This shared graph has to be generated dynamically.

- In N-gram decoding, how to represent the word graph? Certainly, storing a fully connected word graph inside SearchLevel is not a good choice.

- If using monophones, at the phone level, the subgraphs are just the pronunciation graphs. What if using word-internal or cross-word triphones? In those cases, the phone graphs are complicated and not fixed. They need to be generated dynamically.

CHAPTER 4

EVALUATION

4.1. Baseline Systems

To build a baseline word-internal triphone system, we trained 12-mixture word-internal triphone models from the scratch on the WS'97 training set [28], and rescored using word graphs generated from a bigram language model. The decoding results are shown in Table 1. Our performance was slightly better than HTK's [29] performance under similar conditions. HTK is a commercially available speech recognition system developed by Cambridge University. It is known to have a leading recognition performance. So we felt confident that our acoustic training and word-graph rescoring were fully debugged and working well.

WER	ISIP	HTK
Substitutions	32.3%	32.2%
Deletions	14.7%	14.8%
Insertions	2.6%	2.9%
Overall	49.6%	49.8%

Table 1. A comparison of a word-internal system with a similar HTK system

(a)

WER	ISIP	HTK
Substitutions	31.0%	30.8%
Deletions	10.0%	11.3%
Insertions	4.1%	3.8%
Overall	45.1%	45.9%

(b)

WER	ISIP	HTK
Substitutions	28.5%	29.5%
Deletions	9.2%	9.4%
Insertions	4.0%	3.9%
Overall	41.7%	42.8%

Table 2. Performance (WER) of a cross-word system (a) using word graphs generated from a bigram LM, and (b) using word graphs generated from a trigram LM

We also generated cross-word triphone models, and rescored them using word graphs generated from a bigram and a trigram language model respectively. Detailed results are shown in Table 2. The WER of our cross-word systems are about 1% better than HTK. For the trigram case, we obtained a WER close to 40%, which is the best number we ever obtained on this particular database.

4.2. A Complete Speech Recognition System

The system below was submitted for Hub-5E 2000 evaluation. It included the following features:

- Front-end: standard 39 MFCC (Mel-frequency Cepstrum Coefficients) features.
- Acoustic models: 16-mixture cross-word triphones models trained on 60 hours of SWB-I and 20 hours of English Call Home data. States were tied using phonetic decision trees.
- Language model: a bigram and a trigram backoff language model were provided by Andreas Stolcke at SRI [20]. They were trained from Switchboard, Callhome and Broadcast news. The full trigram LM generated by SRI has 3,246,315 bigrams and 9,966,270 trigrams. They were pruned using SRI's entropy-based method [16] to eliminate bigrams and trigrams with negligible probabilities. The trigram LM we eventually used contained 138K trigrams, 320K bigrams, and 33K unigrams. The bigram LM simply excluded the 138K trigrams.
- Lexicon: we expanded our standard 22K WS'97 test lexicon to include many new words found in SRI's lexicon. This increased the size of the lexicon to over 33K entries.
- Recognition: performed a two-pass decoding. The first pass is word-graph generation using word-internal triphones and a bigram language model; the second pass is rescoring these word graphs using cross-word triphones and a trigram language model.

In the Hub-5E evaluation, our system achieved a 43.4% WER on the SWB part of the dataset. However this number is not competitive with the results from other sites [30]. After the official evaluation, we fixed several bugs mentioned previously, including the one about N-gram histories. We reevaluated our system on the SWB part of the Hub-5E 2000 dataset and decreased the WER to 42.9%. I felt confident that the bug was fixed, and the trigram LM is properly applied into decoding. But 0.5% is not the substantial reduction in WER we expected. One of the reasons for this is that the word graphs we generated appear to have an excessively high word graph error rate — 19.8%. Therefore the language models could not affect much on rescoring those word graphs. To improve the accuracy of word-graph generation, we need to look into the word-graph generation algorithm further. We also need to improve our acoustic features and models. It appears that in the Hub-5E evaluation, our system was much simper than the systems from other sites. For example, the systems from Cambridge University and SRI used more than 5-pass decoding. They also used more techniques in their feature extraction and HMM training. Another fact is that the coverage of the language models we used is not high enough. By comparing with the reference transcriptions, we measured the OOV (Out of Vocabulary) percentages of the unigrams, bigrams and trigrams in the trigram LM we used. For unigrams, the OOV percentage is 0.6%; for bigrams, the OOV percentage is 14.4%; the OOV percentage of trigrams is 64.5%. If a unigram is out of vocabulary and it happens to be the correct word, it surely results in a recognition error since the decoder does not know this word at all. If bigrams or trigrams are OOV, they will be backoff to unigrams or bigrams.

4.3. Production System

We have performed some preliminary experiments on the context-independent network decoding of the production decoder on TIDIGITS task [31]. The network is a small fully connected network composed by numbers “OH, ZERO, ONE, TWO ..., NINE”. In our experiments, the results and path scores of the production network decoding could fully match the prototype system. This shows that the algorithm in the production decoder is functioning correctly. Currently the speed of the production decoder is about twice slower than the prototype system. This is not a surprise since there are overhead for generalization. We are looking at the speed problem and will overcome it in the early stage of design and development.

CHAPTER 5

CONCLUSION AND FUTURE WORK

This thesis introduced the principle of speech recognition; described N-gram decoding and network decoding in detail. The work was based on our ISIP public-domain Speech to Text (STT) systems. The prototype ISIP STT system has been developed into a full-fledged system, including an audio front-end, an HMM training module, and a hierarchical decoder. The decoder is the core of the speech recognition system. It has network decoding (word-graph rescoring), word-graph generation and N-gram decoding etc. functionalities. Particularly, this thesis focused on how to apply N-gram language models into the decoding process. Language model storage, N-gram history handling, search space organization and pruning are the key issues towards an efficient N-gram decoding. Through developing and debugging the decoder, we got deeper understanding about the algorithms and some efficiency issues. With the enhancements we made and fixing some crucial bugs, the system performance has been improved and achieved the best number we got in internal experiments. Our systems are competitive to the HTK systems under similar conditions. For example, a WER of 41.7% was achieved using cross-word models and rescoring word graphs generated by a trigram language model.

We are confident about our implementations of the decoding algorithms. To improve the overall performance of our system, we need to introduce new techniques into the recognition systems, such as gender dependent models, VTLN (Vocal Tract Length Normalization) [32], maximum likelihood linear regression (MLLR) [33] for speaker

adaptation. Improving the speech features, acoustic models and language models can result in significant improvement on performance.

Being extensible and generalizable are also our concerns to a speech recognition system. Therefore we started to develop our efficient and extensible production decoder. The production decoder is based on the highly flexible ISIP foundation classes. We have incorporated the context-independent network decoding function into the production decoder. Preliminary experiments showed that the production network decoding could fully match the prototype decoding. The production decoder is more generalized, and easier to extend. As the next step, we will improve the speed of network decoder in the production decoder. Then we will work on context-dependent network decoding, N-gram decoding and word-graph generation. In short, there is still a lot of work to do.

APPENDIX A

Table 1.IPA and ARPABET Representations of Phonemes

ARPABET	Examples	IPA
aa	lOck	ɑ
ae	bAt	æ
ah	bUt	ʌ[1]
ao	bOUght	ɔ
aw	cOW	ɑʊ
ax	About	ə
ay	bUY	aI
b	Bet	b
ch	CHurch	c͡
d	Debt	d
dh	THat	ð
eh	bEt	e
el	battLE	ɬ
en	buttON	n
er	bIRd	ɚ
ey	bAIt	eI
f	Fat	f
g	Get	g
hh	hello	h
ih	bIts	ɪ[1]
iy	bEAt	i

ARPABET	Examples	IPA
jh	Judge	j͡
k	Kit	k
l	Let	l
m	Met	m
n	Net	n
ng	siNG	ŋ
ow	bOAt	o
oy	bOY	ɔi
p	Pet	p
r	Rent	ɹ
s	Sat	s
sh	SHut	ʃ
t	Ten	t
th	THree	θ
uh	bOOk	*U*
uw	tOO	ʊ
v	Vat	v
w	Wit	w
y	You	j
z	Zoo	z
zh	pleaSure	ʒ

REFERENCES

[1] W.J. Ebel and J. Picone, "Human Speech Recognition Performance on the 1994 CSR Spoke 10 Corpus," Proceedings of the Spoken Language Systems Technology Workshop, pp. 53-59, Austin, Texas, USA, January 1995.

[2] L. Rabiner, *Fundamental of Speech Recognition*, Prentice Hall, 1993.

[3] The International Phonetic Association, *http://www.arts.gla.ac.uk/IPA/ipa.html*

[4] J. Zhao, Phonemes, *http://www.isip.msstate.edu/projects/switchboard/doc/education/phone_comparisons/*, Institute for Signal and Information Processing, Mississippi State University, July 1999.

[5] "Alphadigit v1.0," *http://cslu.cse.ogi.edu/corpora/alphadigit/*, Center for Spoken Language Understanding, Oregon Graduate Institute of Science and Technology, USA, 1997.

[6] F. Jelinek, *Statistical Methods for Speech Recognition*, Massachusetts Institute of technology, 1997.

[7] J. Picone, "Signal Modeling Techniques in Speech Recognition," *IEEE Proceedings*, Vol. 81, no. 9, pp. 1215-1247, September 1993.

[8] V. Mantha, R. Duncan, J. Zhao, J. Picone, Implementation and Analysis of Speech Recognition Front-ends, *Proceedings of the IEEE Southeastcon*, Lexington, Kentucky, USA, March 1999.

[9] V. Valtchev, *Discriminative Methods in HMM-based Speech Recognition*, Ph. D. Thesis, University of Cambridge, UK, 1995.

[10] ISIP public domain Speech-to-text system, *http://www.isip.msstate.edu/projects/speech/*, Institute for Signal and Information Processing, Mississippi State University

[11] N. Deshmukh, A. Ganapathiraju and J. Picone, "Hierarchical Search for Large Vocabulary Conversational Speech Recognition," IEEE Signal Processing Magazine, Vol. 1, no. 5, pp. 84-107, September 1999.

[12] R. Sundaram, A. Ganapathiraju, J. Hamaker and J. Picone, "ISIP 2000 Conversational Speech Evaluation System," *Proceedings of the Speech Transcription Workshop*, College Park, Maryland, USA, May 2000.

[13] A. Martin, M. Przybocki, J. Fiscus and D. Pallett, “The 2000 NIST Evaluation for Recognition of Conversational Speech over the Telephone,” *Proceedings of the Speech Transcription Workshop*, College Park, Maryland, USA, May 2000.

[14] J. Odell, *The Use of Context in Large Vocabulary Speech Recognition*, Ph.D. Thesis, Cambridge University, 1995.

[15] J. Odell, V. Violative and P. Woodland, “A One-Pass Decoder Design for Large Vocabulary Recognition,” *Proceedings of the DARPA Human Language Technology Workshop*, March 1995.

[16] A. Stolcke, “Entropy-Based Pruning of Backoff Language Models,” *Proceedings of DARPA Broadcast News Transcription and Understanding Workshop*, Lansdowne, Virginia, USA, pp. 270-274, February 1998.

[17] R. Rosenfeld, “*Adaptive Statistical Language Modeling*,” Ph.D thesis, Carnegie Mellon University, Pittsburgh, PA, USA, April. 1994.

[18] Mosur K. Ravishankar, *Efficient Algorithms for Speech Recognition*, Ph.D. Thesis, Carnegie Mellon University, 1996.

[19] A. Martin, M. Przybocki, J. Fiscus and D. Pallett, “The 2000 NIST Evaluation for Recognition of Conversational Speech over the Telephone,” *Proceedings of the Speech Transcription Workshop*, College Park, Maryland, USA, May 2000.

[20] SRI Language Modeling toolkit, *http://www.speech.sri.com/projects/srilm*, SRI inter-national Corporation, CA.

[21] M. Woszczyna, M. Finke, “Minimizing Search Errors Due to Delayed Bigrams in Real-time Speech Recognition Systems,” *Proceedings of Acoustics, Speech, and Signal Precessing Conference (ICASSP-96)”*, Vol. 1, pp. 137-140, 1996

[22] S. Ortmanns, H. Ney and A. Eiden, “Language Model Look-ahead for Large Vocabulary Speech Recognition”, *Proceedings of the Fourth International Conference on Spoken Language Processing*, pp. 2095-2098, October 1996.

[23] N. Deshmukh, A. Ganapathiraju and J. Picone, “Hierarchical Search for Large Vocabulary Conversational Speech Recognition,” *IEEE Signal Processing Magazine*, Vol. 16, no. 5, pp. 84-107, September 1999.

[24] A. Ganapathiraju, N. Deshmukh, and J. Picone, “ISIP Public Domain LVCSR System,” *Proceedings of the Hub-5 Conversational Speech Recognition (LVCSR) Workshop*, Linthicum Heights, Maryland, USA, June 1999.

[25] “ISIP Software Documentation,” *http://www.isip.msstate.edu/projects/speech/education/tutorials/isip_env/*, Institute for Signal and Information Processing, Mississippi State University, September 2000.

[26] A. Martin, J. Fiscus, M. Przybocki and B. Pallett, “The Evaluation: Word Error Rates and Confidence Analysis,” *Proceedings of the 9th Hub-5 Conversational Speech Recognition Workshop*, Linthicum Heights, Maryland, USA, September 1998.

[27] J. Zhao, A. Ganapathiraju, V. Mantha, J. Hamaker, and J. Picone, “Enhanced Acoustic Modeling For A Free Speech To Text System,” quarterly report for Department of Defense, Institute for Signal and Information Processing, Mississippi State University, March 31, 2000.

[28] LVCSR Workshop at Johns Hopkins University, *http://www.clsp.jhu.edu/1997/*.

[29] S. J. Young, “The HTK Hidden Markov Model Toolkit: Design and Philosophy,” CUED/F-INFENG/TR.152, Cambridge University, 1994.

[30] “NIST March 2000 Hub-5 Benchmark Test Results,” *ftp://jaguar.ncsl.nist.gov/lvcsr/mar2000/hub5_scores_20000511/readme.htm*, National Institute of Standards and Technology, USA, May 2000.

[31] J. Hamker, “ISIP Digit Recognition Tutorial,” *http://www.isip.msstate.edu/projects/speech/education/tutorials/asr_tidigits/*, August, 2000.

[32] T. Kamm, G. Andreou and J. Cohen, “Vocal Tract Normalization in Speech Recognition Compensation for Systematic Speaker Variability,” Proceedings of 15th Annual Speech Research Symposium, pp. 175-178, CLSP, Johns Hopkins University, Baltimore, Maryland, 1995.

[33] M. J. F. Glales, D. Pye and P. C. Woodland, “Variance Compensation within the MLLR framework for robust speech Recognition and Speaker Adaptation,” Proceeding of International Conference on Spoken Language Processing (ICSLP) 1996, Vol. 3, pp. 1832-1835, Philadelphia, PA, USA, October 1996.