METHODOLOGIES FOR LANGUAGE MODELING AND SEARCH IN CONTINUOUS SPEECH RECOGNITION

Neeraj Deshmukh
Dept. of Electrical Engineering
Boston University
Boston, MA 02215

Joseph Picone
Instt. for Signal & Information Processing
Mississippi State University
MS State, MS 39762

Abstract **- Automatic speech recognition has made significant strides from the days of recognizing isolated words. Today state-of-the-art systems are capable of recognizing tens of thousands of words in complex domains such as newspaper correspondence and travel planning. A major part of this success is due to recent advances in language modeling and search techniques that support efficient, sub-optimal decoding over large search spaces. The benefit from focusing a recognition system on a particular domain has motivated a steady progression from static language models towards more adaptive models that consist of mixtures of bigrams, trigrams and long-distance n-grams. Similarly, availability of multiple sources of information about the correct word hypothesis has led to the advent of efficient multi-pass search strategies. The result is a powerful pattern-matching paradigm that has applications to a wide range of signal detection problems. Future research in large vocabulary continuous speech recognition will be directed towards developing more efficient means of dynamically integrating such information.**

INTRODUCTION

The aim of continuous speech recognition (CSR) is to provide an efficient and accurate mechanism to automatically transcribe speech into text. As different words are spoken at different times by different people, a statistical approach to CSR appears to be naturally applicable . If a sequence of words

$$W = w_1, w_2, \ldots, w_N \tag{1}$$

is spoken, and if A is the acoustic evidence that is provided to the recognition system to identify this sequence; then the recognizer should decide in favor of the word string $\hat{W}$ which maximizes $p(W/A)$, the probability that the word string W was spoken given that data A was observed:

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

Using Bayes formula this equation reduces to finding $\hat{W}$ such that

$$\hat{W} = \arg\max_{W} p(W)p(A/W). \tag{3}$$

The probability $p(A/W)$ that the data A will be observed if a word sequence W was spoken is given by what is known as the *statistical acoustic model.* The probability $p(W)$ that enumerates the a priori chances of the word sequence W being spoken is determined by the *statistical language model.* Typically, Hidden Markov Models (HMMs) [1] are used to construct the acoustic models, while language models are mostly based on a Markov process. The recognizer computes the likelihood of the observed data using the acoustic models. Scores for various word sequences or hypotheses are generated using the acoustic model scores and the probability of the word sequences which is given by the language model. The hypothesis corresponding to the maximum score is chosen as the correct word sequence.

The number of word sequences is quite large even for a small vocabulary, and the process of scoring the hypotheses is a fairly complicated one. Therefore closed-form solutions that can be obtained using linear-algebraic methods do not exist. A search paradigm needs to be employed to select a solution from numerous alternatives based on some criteria.

In this paper we will concentrate on the language modeling and search aspects of CSR. In the next section we provide a more detailed introduction to the problem of language modeling followed by an overview of various language modeling techniques. Various aspects of the search strategy and some predominant search techniques are discussed next.

STATISTICAL LANGUAGE MODELING

The choice and scope of the language model chosen has a significant influence on the performance of a speech recognition system. A language model is important as it provides constraints on the occurrence of particular words and word sequences. It thus plays a considerable role in determining the search space and hence the appropriate search strategy for the recognition process.

The problem of language modeling becomes computationally expensive for a large vocabulary set. The rules of simple formal grammars are inadequate to provide a sufficient framework for recognition. Real speech is not strictly grammatical and involves awkward phrasing or abbreviated word-forms. These depend on the context of the conversation or assume a corresponding knowledge from the listener. A good language model should be able to incorporate such grammatical constraints, topical dependencies, constraints imposed by the accent and style of the speaker etc. It should also be compact enough to allow a reasonably efficient real-time implementation.

Given different language models, we need a framework to objectively compare them to determine which one is better than the others. Such a goodness criterion is discussed next.

Perplexity

Perplexity [2] is an objective measure of language model quality and derives its roots from information theory. A language source (e.g. a speaker) provides information in speaking a word by removing the uncertainty about the identity of that word. The greater the uncertainty about the next word, more is the information contained in it. A measure of this information is *entropy* which is defined as follows:

Entropy: If a random variable X takes N independent values $x_1, x_2, \ldots, x_N$ with probabilities $p(x_1), p(x_2), \ldots, p(x_N)$, its entropy is given by

$$H(X) = -\sum_{i=1}^{N} p(x_i) \log_2 p(x_i). \tag{4}$$

We can consider the language model to be a random process representing different words $w_1, \ldots, w_T$. Correspondingly we can find the entropy $H(w)$ of this language model. The *perplexity* of this model is given by

$$P = 2^{H(w)}. \tag{5}$$

The value of perplexity depends on the data on which the language model is trained as well as on the test data. It treats words as abstract symbols and does not take into account the acoustic similarity between words. A language model with low perplexity provides some bounds on the performance of the system, as the recognizer has a correspondingly smaller number of equiprobable choices to pick from. However, for acoustically similar words performance of the recognizer suffers irrespective of perplexity. Thus, perplexity is not necessarily related to accuracy and a good language model should be able to satisfy both of these criteria.

Static Language Models

Language models are used by CSR systems to apply various levels of constraints to the recognizer. A *uniform* language model that assigns equal probabilities to all words in the vocabulary does not impose any constraint on the recognizer and therefore is not very useful. We need to find information about the identity of the word w_i given its history i.e. the word sequence $w_1, \ldots, w_{i-1}$ preceding it in the document.

In a statistical model, we can represent the probability of occurrence of the word sequence W in (1) as

$$p(W) = \prod_{i=1}^{N} p(w_i / w_1, \ldots, w_{i-1}). \tag{6}$$

A language model that uses the history of the $n-1$ immediately preceding words to compute the probability of occurrence of the current word is called a *n-gram* model [3].

$$p(W) = \prod_{i=1}^{N} p(w_i / w_1, \ldots, w_{i-n+1}). \tag{7}$$

A value of n greater than 3 is not practicable for implementation except on extremely small vocabularies and hence is typically limited to 2 (bigram model) or 3 (trigram model). A trigram is better than a bigram model in terms of both accuracy and perplexity as it carries more information.

The n-gram model is simple yet powerful [4], but it is static. It uses only the very immediate history of the word and does not depend on or vary with the data being observed. Therefore it is not capable of adapting to the style or topic of the document and cannot exploit these to enhance the probabilities of related words while suppressing those of others. Therefore we need to explore some dynamic models.

Dynamic Language Models

An adaptive or dynamic model improves upon the performance of a static trigram model by changing estimates of word probabilities depending upon the part

of the document observed so far. This is particularly useful if a model trained on data pertaining to a specific domain is used in another domain, as the model can adjust to the new language. Also, if a large language source can be envisioned as consisting of small homogeneous *chunks* or *sublanguages* (e.g. newspaper articles); then an adaptive model trained on the heterogeneous source can exploit the sublanguage structure to improve performance. Some language models that attempt to capture these long-distance linguistic phenomena like topic-dependence are as follows:

Long-distance n-grams: These are similar to conventional n-grams except that they precede the word w_i by j positions [5].

Triggers: A *trigger pair* [6, 7] consists of two word sequences where the occurrence of one changes the probability estimate of the other. Constructing a trigger model involves eliminating all pairs of word sequences that are not significantly related. The effects of several triggers towards the triggered sequence are combined and the trigger information is integrated with the static model in a way that preserves the advantages of both.

A *Maximum Entropy* (ME) algorithm [8, 9] is used to train the trigger-based language model. While the ME approach is intuitively simple, easy to implement to a variety of problems and guaranteed to converge to a solution, it suffers from very high requirements of memory and computation and does not have a well-defined rate of convergence.

Cache Models: Once a word (or word sequence) occurs in a document, its likelihood of recurrence is greatly increased. This tendency is most true of rare words, and reduces as the word becomes more frequent in occurrence. Based on this phenomenon, the last L words (or word sequences) of the document seen so far are stored in a cache. This cache is used to estimate the dynamic unigram, bigram and trigram probabilities [10, 11, 12] and then incorporated with the static model using interpolation techniques. Caches can also be formed based on the number of times w_i already appeared in the history and based on distance i.e. the last time w_i occurred in the history.

Class Grammars: Instead of words, classes of words are taken as the units of the model. The probability of word occurrence is determined by the probability of occurrence of that word class.

Tree-based models: These models [13] generate a binary decision tree from the training data to cluster word histories. Each node of the tree is associated with a state of the language model and each leaf corresponds to a legal word sequence. The tree is constructed using yes/no questions that reduce the uncertainty of predicting the next word at every node and thus minimize the average entropy at every leaf. However, these methods are computationally expensive.

Mixture models: The language model is built as a mixture of several component models, each of which is trained on the n-gram statistics of a particular topic or broad class of sentences. The component models can be combined using either dynamic-weight mixtures at n-gram level [14] or static-weight mixtures at sentence level [15, 16]. The topics can be specified by hand, or can be determined automatically using clustering techniques. Robustness of parameter estimation for mixture components is an important issue here as each component model is trained only on a part of the available data which corresponds to a particular topic.

Other Techniques

There are various other techniques of language modeling that have different properties. For instance, while *context-free* [17] and *unification* [18] models are more realistic, they are computationally cumbersome. On the other hand, *finite state* [19, 20] models that try to model all legal sentences in a single network are constraining but they are not as realistic.

After incorporating both the language model and the acoustic model in the recognition system, the next step is to evaluate the data and search for the best hypothesis. In the next section we discuss this aspect of CSR.

SEARCH IN CSR

A decoding strategy is used to find the most likely word sequence given the language and acoustic models and a spoken utterance. A simple and intuitively obvious search strategy would be to simply enumerate all possible hypotheses and pick the most likely one. However, since the number of possible hypotheses grows exponentially with the length of the word sequence, this *enumerative search* is practical only for super-trivial tasks. For more realistic problems we need to restructure this unrestricted search algorithm so that the recognizer can find a solution in a finite time interval [21]. The hypothesis generating process is optimized by merging common partial hypotheses. The search space is reduced by heuristically pruning away hypotheses with low scores. Applying such transformations to the problem space causes the system to make suboptimal decisions, though this does not seem to affect the accuracy of recognition. External knowledge sources are also employed to improve search efficiency. Two commonly used search algo-

rithms that employ the above techniques are described next.

Viterbi Search

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. The Viterbi search algorithm [22] builds a breadth-first search tree out of this network in the following fashion:

1. If N is the duration of the utterance, N number of state lists S are generated. These lists are initialized by setting the probability of the initial state as 1 and the others 0.
2. For each state $s \in S(t)$

   For each possible transition from s to a state $\acute{s} \in S(t+1)$

   - Compute the transition probability $p(\acute{s}/s)$.
   - If $\acute{s}$ is uninitialized, initialize it with score $p(\acute{s}/s)$ and a backpointer to s.
   - Else update $score(\acute{s})$ only if this transition gives a better score.
3. If $t = N$ backtrack; else go to step 2 with $t = t+1$.

Viterbi search is time-synchronous; i.e. at any stage all partial hypotheses correspond to the same portion of the utterance and hence can be directly compared. However, a complete Viterbi search is impractical for even moderate-sized tasks because of the size of the state space. A Viterbi *beam* search [23, 24, 25, 26] is used to reduce the search space.

In Viterbi beam search only the hypotheses whose likelihood falls within a fixed radius of the most likely hypothesis are considered. It is a dynamic programming technique that exploits the observation that many states in the state lists have zero or near-zero scores and therefore need not be considered towards a solution. The best beam size can be determined empirically or adaptively. The advantage of the dynamic beam heuristics is that it allows the search to consider many good hypotheses in absence of a clearly dominant solution. Conversely, in case of a clear best hypothesis few others need to be maintained. The main problem with this strategy is that the same state occurring in different paths needs to be recomputed every time adding to the computation cost.

Many variations of Viterbi beam search have been proposed to improve upon its performance. The state space can be partitioned into subsets that are subject to different beam widths [27]. If there is more information in the form of a larger number of contextual states a tighter pruning threshold is applied. A *maximum* of path scores may be taken when they merge at word boundaries and a *sum* when the merging is within a word [24]. In another modification, additional pruning is performed at the frame level to evaluate only a few best-scoring states [28]. This pruning is typically done only at the few initial frames as almost 95% of hypotheses are generated here. In very large vocabulary problems, a tree structured network in which the states corresponding to common initial phones are shared by different words can be used [29]. This uses the fact that the uncertainty about the identity of the word is much higher at its beginning than at the end and therefore more computation is required at the initial phones than the later ones.

Stack Decoding

Stack decoding search [30] is a depth-first technique similar to the $A^\star$ search [31] in artificial intelligence. It constructs a search tree from the language model state graph where the states correspond to abstract states in the language and the branches represent transitions between these states. The basic stack decoder paradigm [32, 33] can be summarized as:

1. Pop the best partial hypothesis from the stack
2. Apply acoustic and language model fast matches[1] to shortlist the candidate next word.
3. Apply acoustic and language model detailed matches to candidate words.
4. Choose the most likely next word and update all hypotheses.
5. Insert surviving new hypotheses into the stack.

The $A^\star$ stack decoder efficiently combines all information into a single unified one-pass search, though it suffers from problems of speed, size, accuracy and robustness. However, several variations that use weaker and cheaper initial acoustic and language models to produce a list of likely hypotheses that is later refined using more detailed and expensive models have been proposed that improve on its performance.

N-BEST SEARCH

The optimal N-best decoding algorithm [34] is quite similar to the Viterbi search. However, while Viterbi decoding is inherently 1-best, N-best search finds all hypothesis sequences within the specified beam and keeps track of hypotheses with different histories at each state. It then allows only N top-scoring

[1] Fast matches are computationally cheap methods for reducing the number of word extensions which need to be checked by the more accurate but computationally expensive detailed matches.

hypotheses to propagate to the next state. This state-dependent pruning is independent of the global Viterbi beam threshold.

The sources of information on speech used for recognition purposes can be extremely diverse and are correspondingly associated with different costs in terms of computation and memory requirements. A hypothesis that scores the highest given all these knowledge sources will be an optimal solution to the recognition problem. But this typically requires an impractically large search space. It is advantageous to use a strategy in which the most efficient knowledge sources are used first to generate a list of top N hypotheses. These hypotheses can later be re-evaluated with other, more expensive knowledge sources to arrive at the best hypothesis. N-best search provides an efficient method of integrating different knowledge sources and makes the search process more modular. The scores from different knowledge sources can be combined using weights chosen to minimize the recognition error [35].

The N-best paradigm as described above has the problem of being partial towards shorter hypotheses. In other words, if we consider the probability of error in recognition of a single word being roughly independent of its position in the sentence, then a longer sentence will have more errors and therefore will be pushed down in the rank of correct hypotheses. Thus an exact N-best search will require a very large value of N to find the correct answer for a long sentence.

A number of modifications have been proposed to overcome this problem and to make N-best search more accurate and efficient. These modifications allow for some approximations to generate the list of sentences with much less computation. Such approximations are justified as long as the correct hypothesis is assured to be in this list. Even if it does not hold a very high rank in this preliminary list, the correct hypothesis can be found later by rescoring on other knowledge sources.

Lattice N–Best Algorithm

An initial pass of the recognition system is used to build a lattice of word (or phoneme or syllable etc) hypotheses which is searched through by subsequent passes to generate the correct hypothesis. A time-synchronous one-best forward-pass search algorithm is used within words and at each frame all the theories and their respective scores are stored in a traceback list. The best score at this frame is sent forward along with a backpointer to the saved list [36]. The N-best sentences are obtained by recursive search through this traceback list. This algorithm is extremely fast but often underestimates or misses high-scoring hypotheses.

A progressive search [37] can be used to avoid this problem. Here a lattice of all sentence hypotheses is maintained instead of evaluating independent sentence hypotheses. This lattice is treated as a grammar and used to rescore all the hypotheses.

Word–Dependent N–Best Search

This algorithm differentiates between hypotheses on the basis of the previous word instead of the whole preceding sequence [36]. The probability for each of the different preceding words is stored within the word at each state. At the end of the word the score for each hypothesis and the name of the previous word are recorded. A recursive traceback is used at the end of the sentence to derive the list of the most likely sentences.

Forward–Backward Search

Forward-backward search algorithms use an approximate time-synchronous search in the forward direction to facilitate a more complex and expensive search in the backward direction [36, 38, 39, 40]. This generally results in speeding up the search process on the backward pass as the number of hypotheses to be explored is greatly reduced by the forward search. A simplified acoustic or language model is used to perform a fast and efficient forward-pass search in which the scores of all partial hypotheses that fall above a pruning beamwidth are stored at every state. Then a normal within-word beam search is performed in the backward direction to generate the N-best hypotheses list. The backward search scores high on a hypothesis only if there also exists a good forward path leading to a word-ending at that time.

Since the forward-backward search allows use of different models on the two passes, a complex model can be used on the backward pass to come up with extremely accurate results [41]. The forward scores, though not exact, are good enough estimates of the word end scores and can be further modified by normalization relative to the highest score in each frame. The time-synchronous nature of both passes allows them to have different normalized scores without loss of accuracy.

Forward-backward search algorithms have greatly facilitated real-time handling of large-scale tasks. The backward pass search is fast enough to be performed without any perceptible delay after the forward search. The forward search can be made more approximate and hence efficient as the scores need not be very accurate on the forward pass.

A variation of the forward-backward N-best search is a *tree-trellis based fast search* algorithm [42] that uses a modified Viterbi beam algorithm in the forward pass and an $A^\star$ stack decoder search on the backward pass. The partial hypothesis map prepared in the forward trellis search is used by the backward search to estimate the incomplete portion of the partial hypothesis.

CONCLUSION

The techniques described here have been incorporated to some extent into most modern-day large vocabulary systems. Use of such techniques in acoustic and language modeling have resulted in real-time implementation of systems capable of recognizing over 40,000 words in modest amounts of general purpose hardware [27, 33, 37].

However, these advances still fall far short of demands of operational systems. For example, the same technology performs at a 50% word error rate on conversational speech collected over the telephone. Even under laboratory conditions, such technology is unable to handle many conversational speech phenomena (referred to as dysfluencies). For example, one such common phenomena that is poorly represented in today's systems is called false-start ("Please give me, uh, no, just a second, ok, please give me a red one.")

Our future research will be oriented towards exploring alternative language models that improve performance by providing the recognizer with more specific context, yet significantly reduce the search space. We will focus on dynamic language models that accurately incorporate the long-distance effects on word occurrence. Specifically, we will try to develop improved adaptive mixtures of trigrams and long-distance n-grams.

BIBLIOGRAPHY

[1] L. R. Rabiner, "A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition", *Proceedings IEEE*, Vol. 77, No. 2, pp. 257-285, February 1989.

[2] F. Jelinek, R. L. Mercer and S. Roukos, "Principles of Lexical Modeling for Speech Recognition", from *Advances in Speech Signal Processing*, edited by S. Furui and M. M. Sondhi, pp. 651-699, Marcel Dakker Inc., 1992.

[3] L. R. Bahl, F. Jelinek and R. L. Mercer, "A Statistical Approach to Continuous Speech Recognition", *IEEE Transactions on PAMI*, 1983.

[4] F. Jelinek, "Up From Trigrams!", Eurospeech 1991.

[5] X. D. Huang, F. Alleva, H. W. Hon, M. Y. Hwang, K. F. Lee and R. Rosenfeld, "The SPHINX-II Speech Recognition System: An Overview", *Computer, Speech and Language*, 1992.

[6] R. Rosenfeld and X. D. Huang, "Improvements in Stochastic Language Modeling", *Proceedings DARPA Speech and Natural Language Workshop*, February 1992.

[7] R. Rosenfeld, "A Hybrid Approach to Adaptive Statistical Language Modeling", *Proceedings DARPA Human Language Technology Workshop*, pp. 76-81, March 1994.

[8] R. Rosenfeld, "Adaptive Statistical Language Modeling: A Maximum Entropy Approach", *Ph.D. Thesis Proposal*, Carnegie Mellon University, September 1992.

[9] R. Lau, R. Rosenfeld and S. Roukos, "Trigger-Based Language Models: A Maximum Entropy Approach", *Proceedings ICASSP*, Vol. 2, pp. 45-48, April 1993.

[10] J. Kupiec, "Probabilistic Models of Short and Long Distance Word Dependencies in Running Text", *Proceedings ARPA Workshop on Speech and Natural Language*, pp. 290-295, February 1989.

[11] F. Jelinek, B. Merialdo, S. Roukos and M. Strauss, "A Dynamic LM for Speech Recognition", *Proceedings ARPA workshop on Speech and Natural Language*, pp. 293-295, 1991.

[12] R. Kuhn and R. de Mori, "A Cache Based Natural Language Model for Speech Recognition", *IEEE Transactions on PAMI*, Vol. 14, pp. 570-583, 1992.

[13] L. Bahl, P. F. Brown, P. V. de Souza and R. L. Mercer, "A Tree-Based Statistical Language Model for Natural Language Speech Recognition", *IEEE Transactions on ASSP*, Vol. 37, No. 7, pp. 1001-1008, 1989.

[14] R. Kneser and V. Steinbiss, "On the Dynamic Adaptation of Stochastic LM", *Proceedings ICASSP*, Vol. 2, pp. 586-589, April 1993.

[15] R. Iyer, M. Ostendorf and J. R. Rohlicek, "An Improved Language Model Using a Mixture of Markov Components", *Proceedings DARPA Human Language Technology Workshop*, pp. 82-86, March 1994.

[16] R. Iyer, "Language Modeling with Sentence-Level Mixtures", *M.S. Thesis*, Boston University, 1994.

[17] H. Ney, "Dynamic Programming Speech Recognition Using A Context-Free Grammar", *Proceedings ICASSP*, pp. 321-324, April 1987.

[18] C. Hemphill and J. Picone, "Robust Speech Recognition in a Unification Grammar Framework", *Proceedings ICASSP*, pp. 723-726, May 1989.

[19] J. K. Baker, "Stochastic Modeling as a Means of Automatic Speech Recognition", *Ph.D. Thesis*, Carnegie Mellon University, 1975.

[20] L. R. Bahl, J. K. Baker, P. S. Cohen, A. G. Cole, F. Jelinek, B. L. Lewis and R. L. Mercer, "Automatic Recognition of Continuously Spoken Sentences from A Finite State Grammar", *Proceedings ICASSP*, pp. 418-421, April 1978.

[21] K. F. Lee and F. Alleva, "Continuous Speech Recognition", from *Advances in Speech Signal Processing*, edited by S. Furui and M. M. Sondhi, pp. 651-699, Marcel Dakker Inc., 1992.

[22] A. J. Viterbi, "Error Bounds for Convolutional Codes and an Asymptotically Optimal Decoding Algorithm", *IEEE Transactions on Information Theory*, Vol. IT-13, pp. 260-269, April 1967.

[23] B. T. Lowerre, "The HARPY Speech Recognition System", *Ph.D. Thesis*, Carnegie Mellon University, 1976.

[24] Y. L. Chow, M. Ostendorf-Dunham, O. A. Kimball, M. A. Krasner, G. F. Kubala, J. Makhoul, S. Roukos and R. M. Schwartz, "BYBLOS: The BBN Continuous Speech Recognition System", *Proceedings ICASSP*, pp. 89-92, April 1987.

[25] K. F. Lee and H. W. Hon, "Large-Vocabulary Speaker-Independent Continuous Speech Recognition", *Proceedings ICASSP*, pp. 123-126, April 1987.

[26] H. Ney, D. Mergel, A. Noll and A. Paeseler, "A Data-Driven Organization of the Dynamic Programming Beam Search for Continuous Speech Recognition", *Proceedings ICASSP*, pp. 833-836, April 1987.

[27] F. Alleva, H. Hon, X. Huang, M. Hwang, R. Rosenfeld and R. Weide, "Applying SPHINX-II to the DARPA Wall Street Journal CSR Task", *Proceedings DARPA Speech and Natural Language Workshop*, pp. 393-398, 1992.

[28] N. Deshmukh, J. Picone and Y. H. Kao, "Efficient Search Strategies in Hierarchical Pattern Recognition Systems", to appear in *Proceedings of 27th IEEE Southeastern Symposium on System Theory*, March 1995.

[29] J. J. Odell, V. Valtchev, P. C. Woodland and S. J. Young, "A One Pass Decoder Design for Large Vocabulary Recognition", *Proceedings DARPA Speech and Natural Language Workshop*, pp. 380-385, 1992.

[30] R. L. Bahl, et al., "Large Vocabulary Natural Language Continuous Speech Recognition", *Proceedings ICASSP*, pp. 465-467, May 1989.

[31] N. J. Nilsson, *Problem Solving Methods in Artificial Intelligence*, McGraw-Hill, New York, 1971.

[32] D. B. Paul, "An Efficient $A^{\star}$ Stack Decoder Algorithm for Continuous Speech Recognition with a Stochastic Language Model", *Proceedings ICASSP*, pp. 405-409, March 1992.

[33] D. B. Paul, "The Lincoln Large-Vocabulary Stack Decoder Based HMM CSR", *Proceedings ICASSP*, pp. 374-379, April 1993.

[34] Y. L. Chow and R. M. Schwartz, "The N-Best Algorithm: An Efficient Procedure for Finding Top N Sentence Hypotheses", *Proceedings DARPA Speech and Natural Language Workshop*, pp. 199-202, October 1989.

[35] A. Kannan, M. Ostendorf and J. R. Rohlicek, "Weight Estimation in N-Best Rescoring", *Proceedings DARPA Speech and Natural Language Workshop*, pp. 455-456, February 1992.

[36] R. M. Schwartz and S. Austin, "Efficient, High-Performance Algorithms for N-Best Search", *Proceedings DARPA Speech and Natural Language Workshop*, pp. 6-11, June 1990.

[37] H. Murveit, J. Butzberger, V. Digalakis and M. Weintraub, "Progressive-Search Algorithms for Large-Vocabulary Speech Recognition", *Proceedings DARPA Human Language Technology Workshop*, March 1993.

[38] L. Nguyen, R. Schwartz, F. Kubala and P. Placeway, "Search Algorithms for Software-Only Real-Time Recognition with Very Large Vocabularies", *Proceedings DARPA Human Language Technology Workshop*, pp. 91-95, March 1993.

[39] L. Nguyen, R. Schwartz, Y. Zhao and G. Zavaliagkos, "Is N-Best Dead?", *Proceedings DARPA Human Language Technology Workshop*, pp. 386-388, March 1994.

[40] J. K. Chen and F. K. Soong, "An N-Best Candidates-Based Discriminative Training for Speech Recognition Applications", *IEEE Transactions on Speech and Audio Processing*, Vol. 2, No. 1, Part II, pp. 206-216, January 1994.

[41] R. Schwartz and S. Austin, "A Comparison of Several Approximate Algorithms for Finding Multiple (N-Best) Sentence Hypotheses", *Proceedings ICASSP*, pp. 701-704, 1991.

[42] F. K. Soong and E. F. Huang, "A Tree-Trellis Based Fast Search for Finding the N Best Sentence Hypotheses in Continuous Speech Recognition", *Proceedings ICASSP*, pp. 705-708, 1991.