THE N-BEST ALGORITHM: AN EFFICIENT AND EXACT PROCEDURE FOR FINDING THE N MOST LIKELY SENTENCE HYPOTHESES

Richard Schwartz and Yen-Lu Chow

BBN Systems and Technologies Corp.
Cambridge MA 02138

ABSTRACT

In this paper we introduce a new search algorithm that provides a simple, clean, and efficient interface between the speech and natural language components of a spoken language system. The N-Best algorithm is a time-synchronous Viterbi-style beam search procedure that is *guaranteed* to find the N most likely whole sentence alternatives that are within a given a "beam" of the most likely sentence. The computation is linear with the length of the utterance, and faster than linear in N. When used together with a first-order statistical grammar, the correct sentence is usually within the first few sentence choices. The output of the algorithm, which is an ordered set of sentence hypotheses with acoustic and language model scores can easily be processed by natural language knowledge sources without the huge expansion of the search space that would be needed to include all possible knowledge sources in a top-down search.

I Introduction

In a spoken language system (SLS) we have a large search problem. We must find the most likely word sequence consistent with all knowledge sources (speech, statistical N-gram, natural language). The natural language (NL) knowledge sources are many and varied, and might include syntax, semantics, discourse, pragmatics, and prosodics. One way to use all of these constraints is to perform a top-down tightly-coupled search that, at each point, uses all of the knowledge sources (KSs) to determine which words can come next, and with what probabilities. Assuming an exhaustive search in this space, we can find the most likely sentence. However, since many of these KSs contain "long-distance" effects (for example, agreement between words that are far apart in the input), the search space can be quite large, even when pruned using various beam-search or best-first search techniques. Furthermore, a top-down search strategy requires that all of the KSs be formulated in a predictive, left-to-right manner. This may place an unnecessary restriction on the type of knowledge that can be used.

The general solution that we have adopted is to apply the KSs in the proper order to constrain the search progressively. Thus, we trade off the entropy reduction that a KS provides against the cost of applying that KS. Naturally, we can also use a pruning strategy to reduce the search space further. By ordering the various KSs, we attempt to minimize the computational costs and complexity for a given level of search error rate. To do this we apply the most powerful and cheapest KSs first to generate the top N hypotheses. Then, these hypotheses are evaluated using the remaining KSs. In the remainder of this paper we present the N-best search paradigm, followed by the N-best decoding algorithm. We give an outline of the proof that the algorithm does, in fact, result in the correct list of sentence hypotheses. Finally, we present statistics of the rank of the correct sentence in a list of the top N sentences using acoustic-phonetic models and a statistical language model.

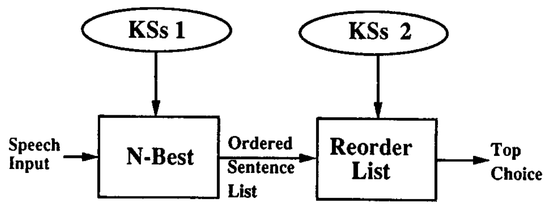
II The N-best Search Paradigm

Figure 1 illustrates the general N-best search paradigm. We order the various KSs in terms of their relative power and cost. Those that provide more constraint, at a lesser cost, are used first in the N-best search. The output of this search is a list of the most likely whole sentence hypotheses, along with their scores. These hypotheses are then rescored (or filtered) by the remaining KSs.

Depending on the amount of computation required, we might include more or fewer KSs in the initial N-best search. For example, it is quite inexpensive to search using a first-order statistical language model, since the number of acoustic and language states is small. Frequently, a syntactic model of NL will be quite large, so it might be reserved until after the list generation. Given a list of hypothesized sentences, each alternative can usually be parsed in turn in a fraction of a second. If the syntax is small enough, it can be included in the initial N-best search, to further reduce the list that would be presented to the remainder of the KSs. We can also use this paradigm in conjunction with high-order statistical language models. While a high-order model frequently provides added power (over a first-order model), the added power may not be commensurate with the large amount of extra computation and storage needed for the search. In this case, a first-order language model can be used to reduce the choice to a small number of alternatives which can then be reordered using the higher-order model.

Besides the obvious computational and storage advantages, there are several other practical advantages of this paradigm. Since the output of the first stage is a small amount of text, and there is no further processing required from the acoustic recognition component, the interface between the speech recog-

CH2847-2/90/0000-0081 $1.00 © 1990 IEEE



KS 1	KS 2
Statistical Grammar	Full NLP
Syntax	Semantics, etc.
Statistical Grammar + Syntax	Semantics, etc.
1st order statistical	Higher-order statistical
•	•
•	•
•	•

Figure 1: The N-best Search Paradigm. The most efficient knowledge sources, KS1, are used to find the N Best sentences. Then the remaining knowledge sources, KS2 are used to reorder the sentences and pick the most likely one.

nition and the other KSs is trivially simple, while still optimal. As such this paradigm provides a most convenient mechanism for integrating work in a modular way. This high degree of modularity means that the different component subsystems can be optimized and even implemented separately (both hardware and software). For example, the speech recognition might run on a special-purpose array processor-like machine, while the NL might run on a general purpose host.

III The N-Best Decoding Algorithm

The optimal N-Best decoding algorithm is, in spirit, quite similar to the time-synchronous Viterbi decoder that is used quite commonly. However, it differs in what it must compute and in its implementation. It must compute probabilities of word-sequences rather than state-sequences, and it must find *all* such sequences within the specified beam. The basic idea is to keep separate records for theories with different word sequence histories. Each path is marked with an identifier that represents the complete sequence of words up to this point (the history). When two or more paths come to the same state at the same time, we check whether there is already an existing path at that state with the same history. If there is, we add the probability for the two paths. Otherwise, we create a new path. When all paths for a state have been created, we reduce the number of paths by keeping up to a specified maximum number N of theories whose probabilities are within a threshold of the probability of most likely word sequence at that state. Note that this state-dependent threshold is distinct from and smaller than the global beam search threshold.

Since probabilities for different word sequences are kept distinct, it is easy to see that any word sequence hypothesis that reaches the end of the sentence has an accurate score. This score is the conditional probability of the observed acoustic sequence given this word sequence. Of course, since the number of possible word sequences grows exponentially, we must use a pruning algorithm to reduce it to the desired number. The interesting question is whether one can prove that all of the word sequences with probabilities greater than the threshold will end up in the list with the correct scores.

Algorithm Optimality

There have been two recent papers that deal with the topic of finding more than one answer for the whole sentence [1, 2]. However, both of these papers are based on the Viterbi algorithm. That is, when two paths for the same word sequence come to the same state, the probability is computed as the maximum of the two paths rather than the sum. Thus these algorithms find the most likely sequence of states rather than the most likely sequence of words. More importantly though, the alternative answers are constrained by the segmentation and traceback of the most likely answer. Since the segmentation of the sentence into words often depends on the words chosen, the answers found in this way are not, in fact, the best N answers. In fact, we have found in the past that this approximation is quite severe. In [2], the exact algorithm for the word sequences corresponding to the best state sequences is mentioned, but is not used, due to the computational requirements. The results given in [2] using a statistical bigram grammar of perplexity 124 show that approximately one third of the sentences that are not recognized correctly on the first choice have the correct answer within the top 10 choices found by the approximate algorithm. As will be seen in the next section, with the exact algorithm used here, for a similar statistical grammar, about 90% of the sentences that are not recognized correctly on the first choice have the correct answer within the top 10 choices found by the approximate algorithm, and about 97% are within the top 24 choices. It should be mentioned that these tests have been performed on different speech corpora, with different acoustic and language models, making direct comparisons difficult.

It should be clear that the algorithm used here would result in the exact solution for all of the possible answers for a given utterance. It is harder to see that the algorithm that finds the N-Best answers within a threshold of the best answer, in fact does so. The proof (which is not included here in its entirety) relies on the fact that the beamwidth at each state is very large – typically on the order of 10^{15}. Possible errors could occur when we should be adding two paths for the same word sequence together, but one or both of them is ignored because its score is more than 10^{15} below the best score at the state. However, if the larger of the two path probabilities was much above the threshold – say 10 times the threshold (still 10^{14} below the best score) – then the error due to ignoring the lower score is insignificant. If both are below the threshold, then when added, they can at most be twice the threshold – still quite low. Even if this happened in every frame of an utterance – an extremely unlikely event – the effect on the score would be small compared to the state beamwidth.

The result is that the algorithm will correctly detect and score all theories that are above the threshold by one order of magnitude. However, the score of theories that are within the last order of magnitude of the final beam may be slightly underestimated. This means that the state beamwidth should be one order of magnitude larger than needed, and the theories within the last order of magnitude can be ignored. When a hard limit of N is placed on the theories at each state, the effective beamwidth at that state could decrease. In this case, we must again include any theories that are within one order of magnitude below the Nth theory at the state to ensure that the final result is correct.

Implementation

This algorithm requires (at least) N times the memory for each state of the hidden Markov model. However, this memory is typically much smaller than the amount of memory needed to represent all the different acoustic models. We assume here, that the overall "beam" of the search is much larger than the "beam at each state" to avoid pruning errors. In fact, for the first-order grammar, it is even reasonable to have an infinite beam, since the number of states is determined only by the vocabulary size.

At first glance, one might expect that the cost of combining several sets of N theories (from preceding states) into one set of N theories at a state might require computation on the order of N^2. However, we have devised a "grow and prune" strategy that avoids this problem. At each state, we simply gather all of the incoming theories. At any instant, we know the best scoring theory coming to this state at this time. From this, we compute a pruning threshold for the state. This is used to discard any theories that are below the threshold. At the end of the frame (or if the number of theories gets much too large), we reduce the number of theories using a *prune and count* strategy that requires no sorting. While this would theoretically still require computation on the order of N, it only accounts for a part of the total computation. We find, empirically, that the overall computation increases with $\sqrt{N}$, or slower than linear. This makes it practical to use somewhat high values of N in the search.

IV Rank of the Correct Answer

Whether the N-best search is practical depends directly on whether we can assure that the correct answer is found reliably within the list that is created by the first stage. (Actually, if all the remaining KSs have binary scores, that is they either accept or reject a sentence, then the search is sufficient as long as there is one answer that is acceptable, since the system could never choose the lower scoring correct answer in this case.) It is possible that when the correct answer is not the top choice, it might be quite far down the list, since there could be exponentially many other alternatives that score between the highest scoring answer and the correct answer. Whether this is true depends on the power of the acoustic-phonetic models and the statistical language model used in the N-best search. Therefore we have accumulated statistics of the rank of the correct sentence in the list of N answers for two different language models: a first-order statistical class grammar (perplexity 100) [3], and no grammar (perplexity 1000). The first-order class grammar constrains the probabilities of all words in the same class to be the same, and therefore can be estimated from a small amount of training data. The experiment was performed on the speaker-dependent portion of the DARPA 1000-Word Resource Management speech corpus [4], using the BBN BYBLOS Continuous Speech Recognition System [5]. The test includes a total of 215 sentences from 12 speakers.

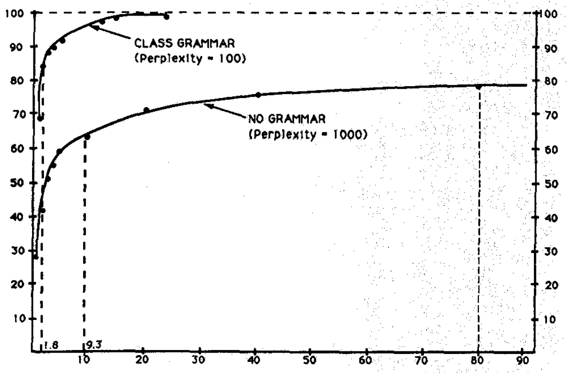


Figure 2: Cumulative Distribution of Rank of Correct Sentence. For the statistical class grammar, 99% of the sentences were recognized exactly within the top 24 choices.

Figure 2 plots the cumulative distribution of the rank for the two different language models. The distribution is plotted for sentence N up to 100. We have also marked the average rank on the distribution. The average rank of the correct answer was 9.3 for no grammar, and the correct answer is not on the list at all about 20% of the time. However, when we use the statistical class grammar, which is a fairly weak grammar for this domain, we find that the average rank is 1.8, since most of the time, the correct answer is within the first few choices. In fact, for this test of 215 sentences, 70% of the sentences were correct on the first choice, while 99% of the sentences were found within the 24 top choices. It is also noteworthy that the acoustic model used in this experiment is an earlier version (that does not model coarticulation between words or use smoothing of poorly trained models) that results in twice the word error rate of the most recent models. This means that the likelihood that the correct answer will be found within a short list of sentences could be even higher than shown here when the better acoustic models are used.

To illustrate the types of lists that are generated we show below a sample N-best output. In this example, the correct answer is the fifth one on the list.

Example of N-best Output
Answer:

Set chart switch resolution to high.

Top N Choices:
Set charts which resolution to five.
Set charts which resolution to high.
Set charts which resolution to on.
Set chart switch resolution to five.
Set chart switch resolution to high. (***)
Set chart switch resolution to on.
Set charts which resolution to the high.
Set the charts which resolution to five.

V Other Applications for N-Best Algorithm

We have, so far, found two additional applications for the N-Best algorithm. The first is to generate alternative hypotheses for discriminative training algorithms. Typically, alternatives must be generated using a fast match procedure, or by using overall statistics of typical errors. Instead, we can generate all the actual alternatives that are appropriate to each particular sentence. This application is discussed in another paper elsewhere in these proceedings [6].

A second application for the N-best algorithm is to generate alternative sentences that can be used to test overgeneration in the design of spoken language systems. Typically, to reduce overgeneration, one generates random sentences using the NL model, examines each sentence to determine whether it makes sense, and changes the grammar to eliminate bad sentence generation. One problem with this is that many of the word sequences generated this way would never, in fact, be presented to a NL system by any reasonable acoustic recognition component. Thus, most of the work may be spent on fixing problems that don't actually occur in a spoken language system. If, instead, we generate N-best lists from a real acoustic recognition system, then we can ask the NL system to parse all the sentences that are known to be wrong. Typically the NL system will reject most of these, and we only need to look at those few that were accepted, to determine whether they should have been.

VI Conclusion

We have presented a new algorithm for computing the top N sentence hypotheses for a hidden Markov model recognition system. Unlike previous algorithms, this one is guaranteed to find the most likely scoring hypotheses with essentially constant computation time. In experiments using a first-order statistical language model, the average rank of the correct answer was 1.8 and was within the first 24 choices 99% of the time. This new algorithm makes possible a simple and efficient approach to integration of several knowledge sources, in particular the integration of complex natural language knowledge sources in spoken language systems. In addition there are other useful applications of the algorithm.

Acknowledgement

This work was supported by the Defense Advanced Research Projects Agency and monitored by the Office of Naval Research under Contract No. N00014-89-C-0008.

REFERENCES

[1] C.H. Lee and L.R. Rabiner (1989) "A Frame-Synchronous Network Search Algorithm for Connected Word Recognition," *IEEE Transactions on ASSP, Vol. 37, No. 11, Nov. 1989, pp. 1649-1658*

[2] V. Steinbiss (1989) "Sentence-Hypotheses Generation in a Continuous-Speech Recognition System," *Proc. of the European Conf. on Speech Communciation and Technology, Paris, Sept. 1989, Vol. 2, pp. 51-54*

[3] A. Derr, and R. Schwartz, "A Statistical Class Grammar for Measuring Speech Recognition Performance," *Proceedings of the DARPA Speech and Natural Language Workshop, October, 1989*

[4] P. Price, W.M. Fisher, J. Bernstein and D.S. Pallett "The DARPA 1000-Word Resource Management Database for Continuous Speech Recognition," *IEEE Int. Conf. Acoust., Speech, Signal Processing*, New York, NY, April 1988, pp. 651-654.

[5] Chow, Y., M. Dunham, O Kimball, M. Krasner, G.F. Kubala, J. Makhoul, P. Price, S. Roucos, and R. Schwartz (1987) "BYBLOS: The BBN Continuous Speech Recognition System," *IEEE ICASSP-87, pp. 89-92*

[6] Y.L. Chow (1990) "Maximum Mutual Information Estimation of HMM Parameters for Continuous Speech Recognition Using the N-Best Algorithm," *Submitted to ICASSP 1990, Albuquerque, New Mexico*