A Comparison of Several Approximate Algorithms For Finding Multiple (N-BEST) Sentence Hypotheses

Richard Schwartz and Steve Austin

BBN Systems and Technologies
Cambridge, MA 02138
Schwartz@bbn.com

ABSTRACT

We introduce a new, more efficient algorithm, called the Word-Dependent N-Best algorithm for finding multiple sentence hypotheses. The new algorithm is based on the assumption that the beginning time of a word depends only on the preceding word. We compare this algorithm with two other algorithms for finding the N-Best hypotheses: The exact Sentence-Dependent method reported in [1] and a computationally efficient Lattice N-Best method. We show that, while the Word-Dependent algorithm is computationally much less expensive than the exact algorithm, it appears to result in the same accuracy. However, the Lattice method, which is still more efficient, has a significantly higher error rate.

We also demonstrate that algorithms that use Viterbi scoring (i.e. they find the word sequences with the most likely single state sequence) have significantly higher error rates than those that use total likelihood scoring (summed over all state sequences).

This new algorithm has been implemented within a real-time spoken language system using commercially available hardware.

1. Introduction

In a Spoken Language System (SLS) we must use all available knowledge sources (KSs) to decide on the spoken sentence. While there are many knowledge sources, they are often grouped together into speech models, statistical language models, and natural language understanding models. To optimize accuracy, we must choose the sentence that has the highest score (probability) given all of the KSs. This potentially involves a very large search space. The N-Best paradigm for integrating several diverse KSs has been described previously [1, 2]. First, we use a subset of the KSs to choose a small number of likely sentences. Then these sentences are scored using the remainder of the KSs.

In [1] we also presented an efficient speech recognition search algorithm that was capable of computing the N most likely sentence hypotheses for an utterance, given the speech models and statistical language models. However, this algorithm greatly increases the needed computation over that needed for finding the best single sentence. In this paper, we introduce a new approximate technique that dramatically decreases the computation needed for the N-Best search. This algorithm is being used in a real-time SLS [3]. In the remainder of the introduction, we review the exact N-Best search briefly and describe its problems. In Section 2, we describe several approximations to the exact algorithm and in Section 3, we compare their accuracy with that of the exact algorithm. In Section 4, we show that other proposed algorithms that use Viterbi scoring (e.g. [4, 5]) must result in lower accuracy.

The N-Best Paradigm

The basic notion of the N-best paradigm is that, while we must ultimately use all the available KSs to improve recognition accuracy, the sources vary greatly in terms of perplexity reduction and computational complexity. For example, a first-order statistical language model can reduce perplexity by at least a factor of 10 with little extra computation over using no grammar, while applying complete natural language (NL) models of syntax and semantics to all partial hypotheses typically requires much more computation for less perplexity reduction. (Murveit [6] has shown that the use of an efficiently implemented syntax component within a recognition search actually slowed down the search unless it was used very sparingly.) Therefore it is advantageous to use a strategy in which we use the most powerful, efficient KSs first to produce a scored list of all the likely sentences. This list is then filtered and reordered using the remaining KSs to arrive at the best single sentence. Figure 1 illustrates this basic idea. For example, we use acoustic models and bigram language models to find the list of likely sentences, and then rescore and filter this list using higher order models and linguistic natural language models. In addition to reducing total computation, the resulting systems are more modular when we separate radically different KSs.

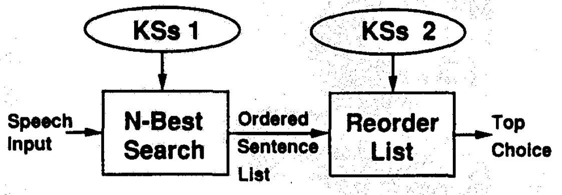

Figure 1: The N-best Search Paradigm. The most efficient knowledge sources, KS1, are used to find the N Best sentences. Then the remaining knowledge sources, KS2 are used to reorder the sentences and pick the most likely one.

Other Uses of the N-Best Paradigm

In addition to its use for efficient combination of multiple knowledge sources within an SLS, we have found other important related uses. We have used the generated sentence lists to provide the alternate hypotheses for corrective training techniques [7]. We have also recently used the N-Best paradigm as a way to quickly measure the effectiveness of new knowledge sources and to combine two radically different recognition technologies, with a minimum of programming and computation. Specifically we combined an HMM recognizer with a recognizer based on Stochastic Segment Models [8] and with one based on Segmental Neural Networks [9]. The basic technique used is that the HMM system first produces an extensive N-Best list for each sentence. Then, the alternate methods merely rescore each of the N sentence hypotheses. Finally, we take the optimal linear combination of the (log) scores from the different knowledge sources and reorder the list of sentence hypotheses. In this way, it is easy to determine whether a new knowledge source contains additional useful information, and it is also trivial to integrate the new knowledge source into the system. Finally, the computation required for the new knowledge source is greatly reduced since it only needs to consider the hypotheses judged plausible by the first

CH2977-7/91/0000-0701 $1.00 © 1991 IEEE

(HMM) system. This reduction in computation is critical for methods that require a large amount of computation (like the stochastic segment model).

The Exact Sentence-Dependent Algorithm

In [1] we presented an efficient time-synchronous algorithm for finding the N most likely sentence hypotheses. This algorithm was unique in that it provided the correct forward probability score for each hypothesis found. The basic idea of the algorithm is that, if two or more theories at a state involve identical sequences of words, we add their scores, since we want the likelihood of the sequence of words summed over all state sequences. Otherwise we keep an independent score for each different preceding sequence of words. We preserve all different theories at each state, as long as they are above the global pruning threshold and within the *state beamwidth* of the best score at the same state. This algorithm *guarantees* finding all hypotheses within a threshold of the best hypothesis. While the proof is not given here, it is easy to show that the inaccuracy in the scores computed is bounded by the product of the sentence length and the pruning beamwidth, which is typically a very small fraction of the score. While the number of theories that are within a threshold of the best theory at a state could theoretically grow exponentially with time (if all theories had about the same score), we find empirically that the number of theories within a threshold remains fairly constant and small. The algorithm was optimized to avoid expensive sorting operations so that it required computation that was less than linear with the number of sentence hypotheses found. In the remainder of the paper, we will refer to this particular algorithm as the Exact or the Sentence-Dependent algorithm.

There is a practical problem associated with the use of this exact algorithm. In cases where we need a large number of hypotheses the computation, which is almost linear in N, becomes excessive. Frequently we compute up to 100 hypotheses for a sentence. In addition, when we examine the different answers found, we notice that many of the different answers are simple one-word variations of each other. Clearly, longer sentences would be expected to have more variations, and thus require a large value of N. Thus, much of the computation is spent finding all combinations of independent variations. In the next section, we present two algorithms that attempt to avoid these problems.

2. Two Approximate N-Best Algorithms

While the exact N-Best algorithm is theoretically interesting, we can generate lists of sentences with much less computation if we are willing to allow for some approximations. (It is important to note that, as long as the correct sentence can be guaranteed to be within the list, the list can always be reordered by rescoring each hypothesis individually at the end.) We present two such approximate algorithms with reduced computation.

Lattice N-Best

The first algorithm will derive an approximate list of the N Best sentences with little more computation than the usual 1-Best search. Within words we use the time-synchronous forward-pass search algorithm [10], with only one theory at each state. We add the probabilities of all paths that come to each state. At each grammar node (for each frame), instead of remembering only the best scoring word, we store all of the different words that arrive at that node along with their respective scores in a traceback list. This requires no extra computation above the 1-Best algorithm. The score for the best hypothesis at the grammar node is passed forward as the basis for future scoring as in the normal time-synchronous forward-pass search. A pointer to the stored list of words and scores is also sent on. At the end of the sentence, we simply search (recursively) through the saved traceback lists for all of the complete sentence hypotheses that are above some threshold below the best theory.

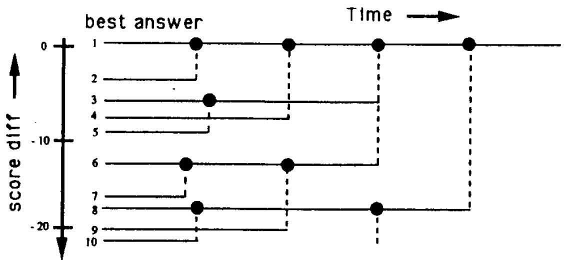

Figure 2: N-Best Theory Traceback. The log-score differences from the best sentence are added to compute the total sentence scores for each alternative.

Figure 2 illustrates several alternate sentence hypotheses stored in the traceback. The locations of alternate word ends are indicated by filled circles. The resulting alternate sentences are shown on the left, ordered by score. Of course the number of sentences represented in this traceback lattice is huge. One algorithm for finding the most likely sentences in the traceback is presented below:

1. Initialize (clear) stack of alternate choices.
2. Initialize accumulated score decrement, s, to 0. Initialize word position, t, to the end of the sentence.
3. Perform traceback computation from t with score s, chaining back through word-ends to produce the next highest scoring sentence. Add this sentence to the list of N-Best sentence hypotheses.
4. At each word boundary along the traceback, add the accumulated score decrement, s, to each of the (negative log) differences between each alternate word score and the current sentence hypothesis. Add these differences to the stack of alternate choices.
5. Pick the alternate with the smallest difference from the best sentence.
6. Perform steps 3 and 4 (recursively) from this point in the sentence until the desired number of hypotheses has been found, or any remaining hypotheses would score too far below the best hypothesis to be of interest.

The alternate sentence hypotheses are produced in order of decreasing total score. This recursive traceback can be performed very quickly. (We typically extract the 100 best answers in a small fraction of a second.) We call this algorithm the Lattice N-Best algorithm since we essentially have a dense word lattice represented by the traceback information. An important advantage of this algorithm is that it naturally produces more answers for longer sentences, since the number of permutations of answers grows exponentially with the length of the utterance.

There is, however, a serious problem with the Lattice N-Best algorithm. It systematically underestimates or completely misses high scoring hypotheses. Figure 3 shows an example in which two different words (words 1 and 2) can each be followed by the same word (word 3). We assume here that the word sequence 2-3 scores better than 1-3. The dark lines for words 1 and 2 show the optimal path for each word when followed by word 3. The gray lines show two suboptimal paths for word 1. Since we allow only one theory at each state within word 3, there is only one best beginning time for word 3, determined by the best boundary between the best previous word (word 2 in the example) and the current word. But, as shown in Figure 3, the second-best theory involving a different previous word (word 1 in the example), would naturally end at a slightly different time. Thus, the best score for the second-best theory would be severely underestimated or lost altogether. Thus, we cannot correctly compute even the second best hypothesis without recomputing the likelihood of the different word sequence.

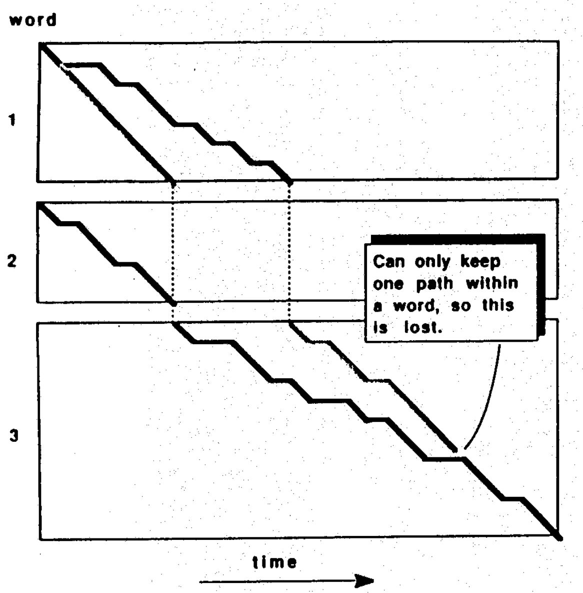

Figure 3: Alternate paths in the Lattice algorithm. The best path for words 2-3 overrides the best path for words 1-3.

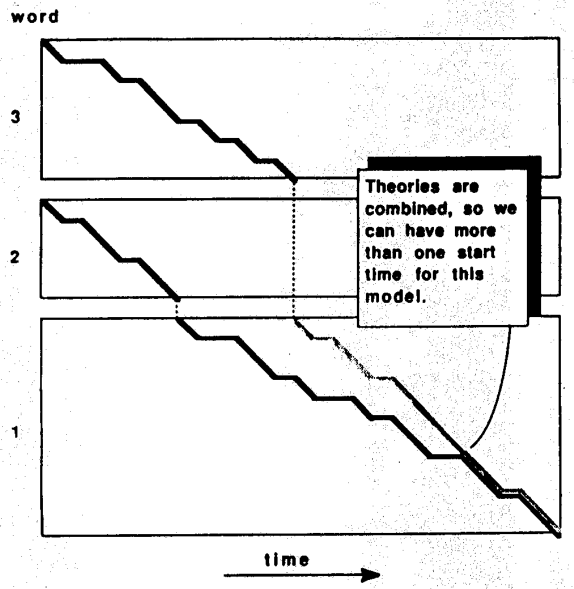
word
3
2
1
Theories are combined, so we can have more than one start time for this model.
time

Figure 4: Alternate paths in the Word-Dependent algorithm. Best path for words 1-3 is preserved along with path for words 2-3.

Word-Dependent N-Best

As a compromise between the exact sentence-dependent algorithm and the lattice algorithm, we devised the Word-Dependent N-Best algorithm. The algorithm is illustrated in Figure 4. We reason that, while the best starting time for a word *does* depend on the preceding word, it probably does *not* depend on any word before that. Therefore, we distinguish theories based on only the previous word rather than on the whole preceding sequence. At each state within the word, we preserve the total probability for each of $n(<< N)$ different preceding words. At the end of each word, we record the score for each previous word hypothesis along with the name of the previous word. Then we proceed on with a single theory with the name of the word that just ended. At the end of the sentence, we perform the recursive traceback (similar to that described above) to derive the list of the most likely sentences. As shown in Figure 4, both sequences are available at the end of word 3.

Like the lattice algorithm, the word-dependent algorithm naturally produces more answers for longer sentences. However, since we keep multiple theories within the word, we correctly identify the second best path (and all others as well).

The computation is proportional to n, the number of theories kept locally, which is typically 3 to 6. The number of local theories only needs to account for the number of possible previous words - not all possible preceding sequences. Thus, while the computation needed is greater than for the lattice algorithm, it is far less than for the sentence-dependent algorithm.

3. Comparison of N-Best Algorithms

We performed experiments to compare the accuracy of the three N-Best algorithms. In all cases, we used a weak first-order statistical grammar based on 100 word classes. The experiments were performed on the speaker-dependent subset of the DARPA 1000-Word Resource Management Corpus [11]. The test set used was the June '88 speaker-dependent test set of 300 sentences. The test set perplexity is approximately 100. To enable direct comparison with previous results, we did not use models of triphones across word boundaries, and the models were not smoothed. We expect all three algorithms to improve significantly when the latest acoustic modeling methods are used.

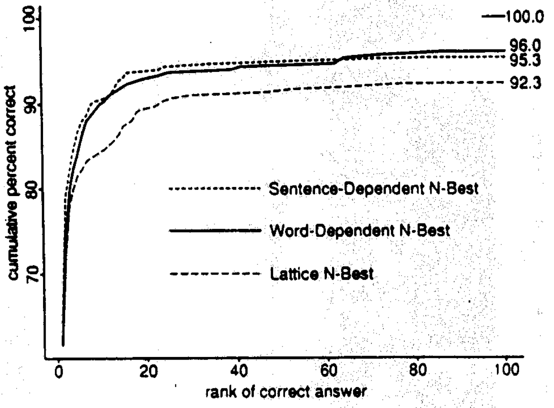

Figure 5: Comparison of the Rank of the Correct Sentence for the Sentence-Dependent, Word-Dependent, and Lattice N-Best Algorithms.

Figure 5 shows the cumulative distribution of the rank of the correct answer for the three algorithms. As can be seen, all three algorithms get the sentence correct on the first choice about 62% of the time. All three cumulative distributions increase substantially with more choices. However, we observe that the Word-Dependent algorithm yields accuracies quite close to that of the Exact Sentence-Dependent algorithm, while the Lattice N-Best is substantially worse. In particular, the sentence error rate at rank 100 (8%) is double that of the Word-Dependent algorithm (4%). Therefore, if we can afford the computation of the Word-Dependent algorithm, it is clearly preferred.

We also observe in Figure 5 that the Word-Dependent algorithm is actually better than the Sentence-Dependent algorithm for very high ranks. This is because the score of the correct word sequence fell outside the pruning beamwidth. However, in the Word-Dependent algorithm, each hypothesis gets the benefit of the best theory two words back. Therefore, the correct answer was preserved in the traceback. This is another advantage that both of the approximate algorithms have over the Sentence-Dependent algorithm.

While the new algorithm presented here is quite efficient, it still increases the computation needed by a significant factor over the 1-Best search. Therefore, we developed a technique called the Forward-Backward Search [3]. The algorithm uses a forward Viterbi search to establish the best score for any path ending with each particular word at each time. This is followed by a backwards version of the Word-Dependent N-Best algorithm, in which only those words that scored well in the forward pass are considered. The result is that the computation of the N-Best list is reduced by a factor of 40. These two algorithms are used in an implementation of a spoken language system that can operate in real time on a commercially available workstation.

4. Suboptimality of Viterbi Scoring

There have been several other algorithms proposed for finding multiple sentence hypotheses. One algorithm ([4, 12]) is quite similar – with some important implementational differences – to the Lattice N-Best algorithm. This algorithm, as described above suffers severely from the problem that it only considers alternate sequences that diverge from the exact boundaries found for the higher scoring paths. The second algorithm, proposed in [5] is the Tree-Trellis algorithm. This algorithm starts with the same forward-pass as used in the Forward-Backward Search. However, it uses a stack search in the backwards direction to find the N-Best word sequences. It must rescore each hypothesis explicitly in the backwards pass in order to guarantee finding the best sentences. Thus the computation needed is proportional to N, which may be a problem for large N. (The Word-Dependent algorithm requires computation that is proportional to the number of local hypotheses only, which is much smaller than N.)

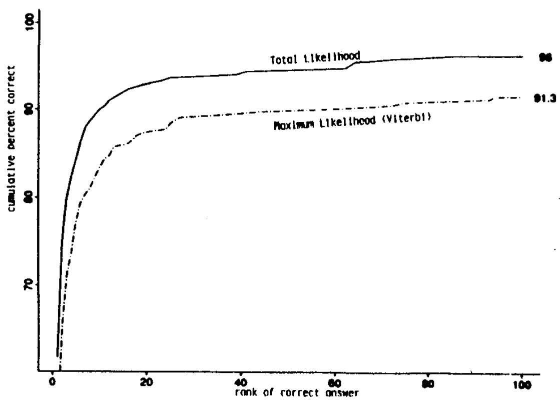

Figure 6: Word-Dependent Algorithm using Viterbi scoring vs forward-likelihood scoring. At a rank of 100, Viterbi scoring misses twice as many sentences.

Both of the algorithms mentioned above are based on Viterbi scoring. That is, they find the word sequence corresponding to the most likely single state sequence. However, we should sum the probability over all possible state sequences that correspond to a word sequence. Figure 6 shows a cumulative distribution of the rank of the correct answer for the two types of scoring within the Word-Dependent N-Best algorithm. At all ranks, there is a 5% increase in the likelihood of finding the correct sentence when we sum the probability over state sequences. While this difference may not be important for rank 1, where the sentence error rate is 38%, it represents a factor of two in the error rate at a rank of 100. Thus, we feel that it is essential, within an N-Best search, to use the total likelihood score for the word-sequence.

5. Conclusion

We have considered several approximations to the exact Sentence-Dependent N-Best algorithm, and evaluated them thoroughly. We show that an approximation that only separates theories when the previous words are different allows a significant reduction in computation, makes the algorithm scalable to long sentences and less susceptible to pruning errors, and does not increase the search errors measurably. In contrast, the Lattice N-Best algorithm, which is still less expensive, appears to miss twice as many sentences within the N-Best choices.

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] Schwartz, R. and Y.L. Chow (1990) "The N-Best Algorithm: An Efficient and Exact Procedure for Finding the N Most Likely Sentence Hypotheses", ICASSP-90, April 1990, Albuquerque S2.12, pp. 81-84. Also in *Proceedings of the DARPA Speech and Natural Language Workshop*, Cape Cod, Oct. 1989.

[2] Young, S. (1984) "Generating Multiple Solutions from Connected Word DP Recognition Algorithms". *Proc. of the Institute of Acoustics, 1984, Vol. 6 Part 4, pp. 351-354*

[3] Austin, S., Schwartz, R., and Placeway, P. (1991) "The Forward-Backward Search Strategy for Real-Time Speech Recognition", elsewhere in these proceedings *IEEE ICASSP-91*, Toronto, Canada, May 1991. Also in *Proceedings of the DARPA Speech and Natural Language Workshop*, Hidden Valley, June 1990

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

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

[6] Murveit, H., "Integrating Natural Language Constraints into HMM-based Speech Recognition". *Proceedings of the ICASSP 90*, April, 1990.

[7] Chow, Y-L., "Maximum Mutual Information Estimation of HMM Parameters Using The N-Best Algorithm for Continuous Speech Recognition". *Proceedings of the ICASSP 90*, April, 1990.

[8] Ostendorf, M., Kannan, A., Kimball, O., Schwartz, R., Austin, S., Rohlicek, R., "Integration of Diverse Recognition Methodologies Through Reevaluation of N-Best Sentence Hypotheses". *Proceedings of the DARPA Speech and Natural Language Workshop*, Monterey, Feb. 1991.

[9] Austin, S., Makhoul, J., Schwartz, R., Zavaliagkos, G., "Continuous Speech Recognition Using Segmental Neural Nets". *Proceedings of the DARPA Speech and Natural Language Workshop* Monterey, Feb. 1991.

[10] Schwartz, R.M., Chow, Y., Kimball, O., Roucos, S., Krasner, M., and Makhoul, J. "Context-Dependent Modeling for Acoustic-Phonetic Recognition of Continuous Speech". *Proceedings of the ICASSP 85*, March, 1985.

[11] Price, P., Fisher, W.M., Bernstein, J., and D.S. Pallett (1988) "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.

[12] Mariño, J. and E. Monte (1989) "Generation of Multiple Hypothesis in Connected Phonetic-Unit Recognition by a Modified One-Stage Dynamic Programming Algorithm", *Proc. of the European Conf. on Speech Communication and Technology, Paris, Sept. 1989, Vol. 2, pp. 408-411*