N-best List Generation using Word and Phoneme Recognition Fusion

Ernest Pusateri[1] *and JM Van Thong*

Compaq Cambridge Research Laboratory
Cambridge, MA 02142 USA
pusateri@mit.edu, jm.vanthong@compaq.com

Abstract

This paper describes an approach for combining phoneme and word recognition to produce an accurate N-best list of hypotheses. We run two decoding threads in parallel. The first performs phoneme recognition, while the other performs word recognition on the same recorded utterance. The output of the word recognition thread is returned as the most likely hypothesis, and the result of the phoneme recognition thread is used to lookup a list of words for the rest of the N-best list. The algorithm is simple to implement and efficient. In our evaluation, we found that this approach has similar performance to the classical lattice-based N-best search methods on isolated word recognition. This method has the potential to improve existing ASR systems or can be used in interactive multi-modal applications.

1. Introduction

Methods for finding multiple sentence or word hypotheses from a spoken utterance, or N-best list, have been thoroughly studied over the past years [1], [2], [3]. The field of applications for N-best lists is vast. They can be used to improve the performance of existing recognizers by narrowing the search space so that other, more computationally intensive algorithms can be applied. Another possible application is for indexing spoken content in video or audio broadcast documents. Most systems use the best hypothesis generated by a large vocabulary word recognizer to produce an indexable transcription. It has been demonstrated that using additional alternate hypothesis from N-best lists may help to improve information retrieval precision [4]. Finally, N-best word lists can be used in interactive applications. A list of alternate choices can be displayed to the user when the system cannot disambiguate between several close word pronunciations.

Classical N-best search methods use a two-pass algorithm: the first pass is a forward Viterbi search, which produces a *word lattice*. During the search, word hypotheses are evaluated based on the acoustic scores of a sub-word unit sequence and on the probability of the previous words (N-grams). Hence, a lattice encodes alternate word hypothesis, possibly with different time segmentations, along with a probability score. At the end of the sentence, a backward search is applied for finding the N-best word hypotheses within the lattice.

One advantage of these methods is that they use the utterance acoustics for building the lattice during the first pass. On the other hand, they consider different time segmentations for the same word, and therefore must apply pruning to remain computationally tractable. As a consequence, only a limited subset of the vocabulary can be extracted (N-best) and re-sorted.

With the performance of speech recognition systems reaching a maximum, information fusion approaches have become more popular [5]. The general idea is that combining different techniques at the acoustic scoring level or decoding level may help to improve the accuracy of systems. In this spirit, we present here an algorithm that uses a fusion approach to compute an N-best list of words. Instead of using a lattice, we combine two recognition threads, one using sub-word units (phonemes), and the other using words. Using a pronunciation distance metric, we then sort the whole vocabulary to create the N-best list. We show that our approach overcomes some of the problems of classical methods and appears to perform as well as these methods on our test set.

The outline of the paper is as follows. In section 2, we describe our approach for combining word and phoneme recognition for resorting a vocabulary to generate an N-best list. In section 3, we present the experimental results. Finally, we present our conclusions and suggestions for future work.

2. Word and phoneme recognition fusion

The algorithm we implemented is intended for isolated word recognition and consists of three main stages, as illustrated in Figure 1. In the first stage, both word and phoneme recognition are performed on the recorded utterance. In the second stage, a phoneme confusion matrix is used with a standard string alignment algorithm to score every word in the vocabulary against the phoneme string returned by the first stage. The vocabulary is then sorted according to these scores. A similar procedure is described in [9]. In the final stage, the word recognition result from the first stage and the sorted vocabulary from the second stage are combined to form an arbitrarily large N-best list.

The two aspects of this algorithm most important to its success are present in the second stage. First is the use of the phoneme confusion matrix. While the word recognition result of the first stage utilizes specific acoustic information about the spoken utterance, the search does not have at its disposal information about the overall performance of the recognizer. It is this additional information that is provided by the phoneme confusion matrix. Indeed, the fact that we achieve comparable performance to the standard lattice-based methods indicates that the information contained in the confusion matrix is at least as useful to N-best list generation as the acoustic information used in those methods.

[1] Ernest Pusateri is now affiliated with the MIT Spoken Language Systems Group, Cambridge, MA 02139, USA.

The second important aspect of the algorithm is the fact that we use the results of phoneme recognition to sort the vocabulary instead of using the results of word recognition. In fact, one could translate the result of word recognition into a phoneme string and then use this to sort the vocabulary.

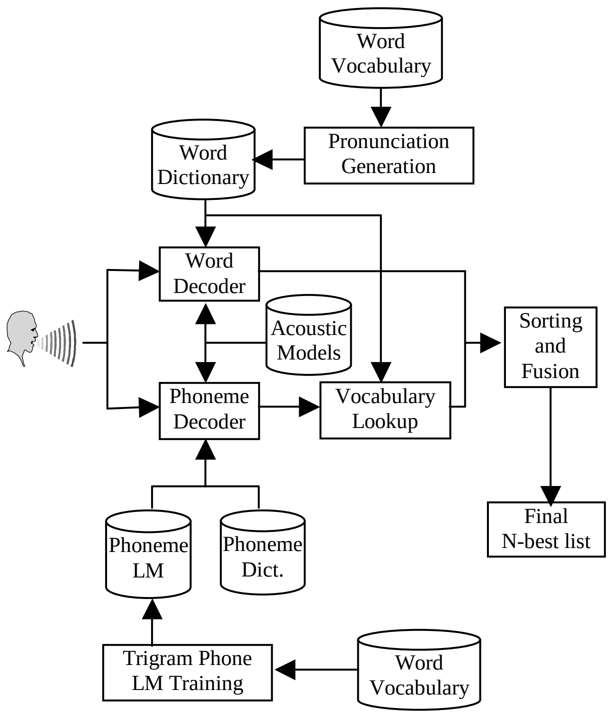

Figure 1: System Overview

However, results given later in this paper prove this to be an inferior approach. If one uses this method, the acoustic representation of the utterance is, in a sense, quantized. From this perspective, each word of the vocabulary is considered a codeword. As always happens with quantization, information is lost in this process.

If one instead uses the result of phoneme recognition to sort the vocabulary, the size of the codebook is dramatically increased. Thus, the negative effects of quantization are reduced, and the performance of the system is increased. While in some applications reduction in quantization comes at a high price, in this algorithm we can achieve these gains with very little compromise in efficiency and no compromise in robustness. The details of our algorithm are described in the sections that follow.

2.1. Phoneme and word recognition

In the first part of the first stage, the speech decoder processes the recorded input speech into a sequence of phonemes. In this part it is important that the decoder has a reasonable degree of freedom to represent the acoustic content of the utterance. Thus, the constraints placed on this decoding pass can be domain specific but should be vocabulary independent. This is accomplished through the use of a phoneme-level language model trained on a large collection of in-domain utterances.

For the second part of the first stage, word decoding is performed. Here the decoder is given much less freedom than in the phoneme recognition pass. The output of the recognizer is constrained to be one of the words in the vocabulary.

2.2. Sorting the vocabulary with a word pronunciation distance metric

The second stage of the algorithm requires that the vocabulary be sorted according to how likely it is that each word was the one spoken, given the phoneme string returned by the first stage. To accomplish this, we define the distance between two words using a word pronunciation distance metric, which is defined as follows.

$$S(p_0, d_0) = 0$$

$$S(p_i, d_j) = \min\left\{\begin{array}{l} S(p_{i-1}, d_{j-1}) + C_{subs}(p_i, d_j) \\ S(p_{i-1}, d_j) + C_{del}(p_i) \\ S(p_i, d_{j-1}) + C_{ins}(d_j) \end{array}\right\}$$

$$S(P, D) = S(p_n, d_n) + LP(p_n, d_n)$$

Where:

- *S(P,D)* is the distance between word P and D; P is the decoded word, and D, the word from the vocabulary,
- $S(p_i,d_j)$ is the score of phoneme string matching up to phoneme p_i of P, and phoneme d_j of D,
- $C_{subs}(p_i,d_j)$ is the cost of substituting phoneme p_i of P with phoneme d_j of D,
- $C_{del}(p_i)$ is the cost of deleting p_i of P,
- $C_{ins}(d_j)$ is the cost of inserting d_j of D,
- $LP(p_n,d_n)$ is the length penalty of decoded phoneme string p_n matching pronunciation d_n.

Computing this metric uses a classical dynamic programming technique [6]. The insertion, deletion, and substitution costs are obtained from a pre-computed phoneme confusion matrix. In addition to the cost of matching string P with D, a length penalty (LP) is applied. This is computed by evaluating the phoneme string length difference between the decoded phoneme string and the pronunciation from the dictionary.

To complete the second stage, the phoneme sequence produced by the recognizer is compared to each word of the vocabulary using the distance metric described above. The word pronunciation distance is then used to sort the whole vocabulary, the most likely word being placed at the top of the list.

2.3. Fusion

In the last stage of the algorithm, the N-best lists generated from phoneme and word recognition are combined to form the final result. We implemented and tested a simple fusion scheme that consists of concatenating the best hypothesis of word recognition with the top N-1 words from the sorted list produced by phoneme recognition. Alternate schemes may be used, but only this one has been tested.

3. Experimental Results

We tested our method on a vocabulary of 9,000 names in which first and last names were aggregated into a single compound word. These names were actor names extracted from several weeks of a TV guide. The test set consisted of 50 spoken queries: 25 names were popular actor names, 15 were picked randomly, and 10 were chosen because they could be easily confused with other names of the vocabulary. Sixteen American or English native speakers recorded the test set using a simple push-to-talk interface, producing a total of 800 utterances.

The recognizer used in these experiments was derived from the CMU Sphinx-3 system [7]. It used 3 emitting-state Gaussian mixture HMMs to model triphones on a vocabulary of 49 phonemes, with 16 Gaussian mixtures per state. Our acoustic models were trained on mel-frequency cepstral coefficients (MFCC) generated from around 100 hours of the 1997 and 1998 Broadcast News corpus provided by LDC [8].

The phoneme trigram language models for phoneme recognition are trained on a set of 10,000 proper names using a dictionary of word pronunciations. For proper names, the pronunciation was either looked up or produced by a lexical tree trained on the 64K most common English words.

The phoneme confusion matrix used to compute the word distance metric was trained using the TIMIT corpus, a collection of 6,300 short, hand-labeled utterances. Training consisted of running phoneme recognition on all of the utterances, then aligning the hypothesized results with the hand transcription. The alignment routine used the same cost for deletion, insertion, and substitution, regardless of the phonemes involved. Alternate approaches are possible for training the confusion matrix, including the use of phoneme classification [9], EM, or genetic algorithms [10].

3.1. Phoneme recognition and word lookup

First we investigate a two-stage algorithm for computing the N-best list without fusion. The recognizer first generated a phoneme string from the utterance. This was done either by using the direct output of phoneme decoding or the pronunciation of the best word hypothesis. In the second stage, an N-best list is generated by sorting the entire vocabulary using the phoneme string from the first stage. This was described in section 2.3.

Table 1 shows the N-best accuracy obtained for various values of N using the phoneme string generated from phoneme or word recognition. We define N-best accuracy as the proportion of utterances of the test set where the correct hypothesis is within the top N words. We see that using word recognition to generate the phoneme string provides drastically better recognition accuracy. However, we also notice that the word accuracy using word recognition does not increase significantly when comparing the 5 best to the 10 best.

N	1	1-5	1-10	1-50	1-200
Phoneme	69.0%	83.4%	87.1%	93.6%	96.9%
Word	85.3%	90.1%	90.6%	93.9%	95.6%

Table 1: N-best accuracy using the phoneme string from phoneme or word recognition for the top 1 to 200 for the 9K words vocabulary.

For these experiments, we used the CMU pronouncing dictionary. For out-of-vocabulary words the pronunciation was generated automatically with letter to sound rules scripts provided with the CMU toolkit [11]. We also ran experiments using a dictionary where hand-generated alternate pronunciations were included. This led to an improvement of about 4% absolute error. These hand-generated pronunciations were inspired by listening to the utterances in the test set. Thus, it is unlikely that any automatic pronunciation generation system could achieve this kind of error rate improvement. However, the experiment does show that this type of two stage recognition system is highly sensitive to word pronunciations, as was shown in [9].

3.2. Phoneme and word recognition fusion

We now investigate the fusion of the two-stage recognition with a single word recognition pass. This was motivated by the observation that the results of these two separate decoding phases were significantly uncorrelated. By combining the results with a simple scheme, we are able to achieve a considerable improvement, as shown in table 2 below. As described in section 2.3, the results are combined by using the result of the word recognition pass as the top word in the N-best list. The rest of the list is filled in using the N-best list obtained from the phoneme hypothesis.

N	1	1-5	1-10	1-50	1-200
Fusion	85.3%	93.6%	95.1%	96.6%	98.0%

Table 2: Fusion scheme between word recognition and the two-stage phoneme recognition, accuracy for the top 1 to 200 for the 9K words vocabulary.

It appears that phoneme recognition allows the decoder some flexibility, while word recognition constrains the search to the word pronunciations in the dictionary. When the word recognition fails to find the best hypothesis, then the two-stage algorithm with phoneme decoding allows recovery, as shown in figure 2.

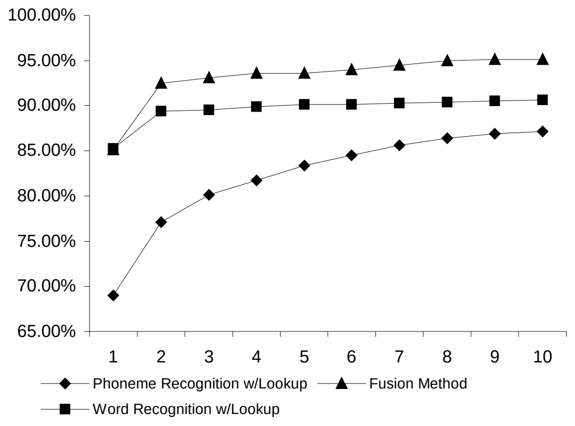

Figure 2: Comparison of the 3 different methods.

In figure 3, we observe that the fusion algorithm performs comparably to the conventional, two-pass, lattice-based N-best search. This leads to two important conclusions. Firstly, the confusion matrix contains information at least as significant to N-best list generation as the information contained in the acoustic lattice generated by standard search algorithms. Exactly how correlated these two sources of information is unclear, however. Secondly, the phoneme string resulting from phoneme recognition provides enough utterance-specific acoustic information to generate an accurate N-best list.

Another possible reason for our algorithm performing comparably to conventional methods is suggested by [12]. The lattice-based method stores only the best segmentation of any particular phoneme. If a hypothesis exists that uses a slightly less probable segmentation of that phoneme, it is not considered. Because of this, sequences like “AH P” and “AH

R P" will not both appear in an N-best list generated using a lattice-based method. Our algorithm does not have this drawback.

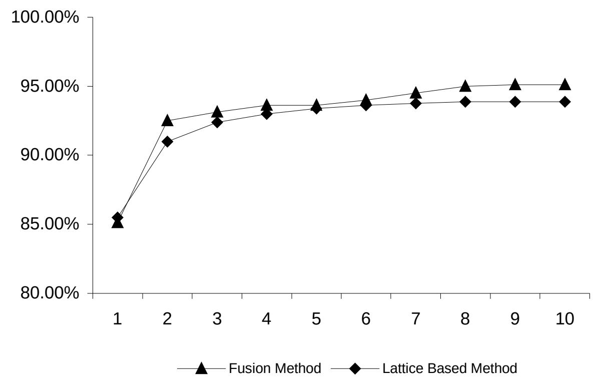

Figure 3: Fusion method versus conventional lattice-based N-best search.

The N-best lists created by this algorithm are different from those generated by conventional methods. Thus, they can potentially be used to augment lists built from word or phoneme based lattices. Other fusion schemes involving the word lattice (instead of the best word hypothesis), and phoneme lattice may lead to further improvements. These information fusion approaches can be exploited in multimedia indexing systems where information retrieval precision prevails. In these systems, the goal is not to produce a transcription, but instead generate enough keywords to allow efficient document retrieval.

4. Conclusions and Future Work

This paper studied a word and phoneme recognition fusion algorithm for generating arbitrary long N-best list of words. The method described performs as well as classical word lattice N-best methods while remaining simple to implement. The results show that we can achieve a precision of about 93.6% within the top 5 on a medium sized vocabulary (9K words). This good performance allows its usage in interactive applications.

In the future, we wish to explore several directions. Confidence scoring may be used at different levels. Phoneme confidence scores can be added to calculation of the word distance, accounting for the uncertainty of both phoneme recognition and pronunciation. Some preliminary experiments show some improvements. We believe that the quality of word pronunciation is critical in order to achieve good results. Adaptive pronunciation methods could help to adjust word pronunciation to common usage. The word distance metric can be improved by carefully scoring multiple phoneme deletions. Finally, the phoneme confusion matrix could be trained in a way that more accurately represents the behavior of the decoder.

5. Acknowledgements

We thank Bhiksha Raj for his help with training the confusion matrices, and for giving us his invaluable insight on the decoding and training processes. We also thank Frederic Dufaux and Shiufun Cheung for providing the test set. Finally, we would like to thank Beth Logan for her insightful comments while reviewing this paper.

6. References

[1] Austin S., Schwartz R., and Placeway P., "The Forward-Backward Search Algorithm", *ICASSP'91, Vol. 1, pp 697, 1991, Toronto, Canada.*

[2] Schwartz R., and Chow Y., "The N-best Algorithm: an Efficient and Exact Procedure for Finding the N Most Likely Sentence Hypotheses", *ICASSP'90, 1990, Albuquerque, USA.*

[3] Tran B-H., Seide F., Steinbiss V., "A Word Graph Based N-best Search in Continuous Speech Recognition", *Proceedings of ICSLP 96.*

[4] Siegler M., "Integration of Continuous Speech Recognition and Information Retrieval for Mutually Optimal Performance". *Ph.D. Thesis, Carnegie Mellon University, 1999.*

[5] Ng K., "Information Fusion for Spoken Document Retrieval". *Proceedings of IEEE Int. Conf. Acoustics, Speech, and Signal Processing, 2000.*

[6] Gusfield D., "Algorithms on Strings, Trees, and Sequences". *Cambridge University Press.*

[7] Placeway P., Chen S., Eskenazi M., Jain U., Parikh V., Raj B., Ravishankar M., Rosenfeld R., Seymore K., Siegler M., Stern R., and Thayer E., "The 1996 Hub-4 Sphinx-3 System". *In Proceedings of the 1997 DARPA Speech Recognition Workshop, Chantilly, Virginia, 1997.*

[8] Stern, R. M., "Specification of the 1996 Hub 4 Broadcast News Evaluation". *In DARPA Speech Recognition Workshop, 1997.*

[9] Coletti P. and Federico M., "A two-stage speech recognition method for information retrieval applications". *Eurospeech, 1999.*

[10] Ferrieux A. and Peillon S., "Phoneme-level indexing for fast and vocabulary-independent voice/voice retrieval". *Proceedings of the ESCA ETRW Workshop: Accessing information in spoken audio, 1999.*

[11] Pagel V., Lenzo K., and Black A. W., "Letter to sound rules for accented lexicon compression". *In ICSLP'98, Sydney, Australia, 1998.*

[12] Fonollosa J. and Batlle E. "Combining Length Restrictions and N-best Techniques in Multiple-Pass Search Strategies". *Eurospeech, 1999.*