

# ESTIMATED RANK PRUNING FOR SPEECH RECOGNITION

Nikola Jevtić Aldebaro Klautau Alon Orlitsky

ECE Department, UCSD  
9500 Gilman Drive  
La Jolla, CA 92093, USA

## ABSTRACT

Most speech recognition systems search through large finite state machines to find the most likely path, or hypothesis. Efficient search in these large spaces requires pruning of some hypotheses. Popular pruning techniques include probability pruning which keeps only hypotheses whose probability falls within a prescribed distance from the most likely one, and rank pruning which keeps only a prescribed number of the most probable hypotheses. Rank pruning provides better control over memory use and search complexity, but requires sorting of the hypotheses, a time consuming task that may slow the recognition process.

We have recently proposed an *estimated rank pruning (ERP)* technique which combines the search-space and memory consumption advantages of rank pruning with the efficiency of probability pruning. Given a prescribed number  $n$  of hypotheses, ERP estimates the probability of the  $n$ -th most likely one and prunes all the less likely ones. In this paper we outline two improvements of ERP: a more accurate rank estimation, and a method for avoiding over-pruning.

## 1. INTRODUCTION

Speech recognition systems commonly model speech as *finite state machines (FSM's)*. The recognition (or decoding) uses a *Viterbi search* to find the most likely path, or *hypothesis* in the FSM. However, in many applications the FSM's are exceedingly large and exhaustive search is not feasible. Sub optimal search algorithms which prune unlikely hypotheses are used instead, e.g., [1, 2].

The most common are the *probability pruning*, or *Viterbi beam search*, algorithms. They are given a prescribed *probability threshold*  $T$ , and, at each step, they extend all previous hypotheses and maintain only the  $T$ -probable ones—those whose log probability is at most  $T$  lower than that of the most likely hypothesis. By contrast, *rank*, or *histogram pruning*, algorithms are given a desired number  $N_{\text{opt}}$  of hypotheses, and at each step, extend all previous hypotheses and maintain only the  $N_{\text{opt}}$  most likely ones. Rank pruning is usually combined with probability pruning so that only the  $T$ -probable among the  $N_{\text{opt}}$  most likely hypotheses are maintained.

Probability pruning is fast, but may maintain an unlimited number of hypotheses, which can take longer to search and require more storage. Rank pruning maintains a fixed number ( $N_{\text{opt}}$ ) of hypotheses, hence better controls memory usage and the size of the search space, but, at every frame, it uses sorting to determine the most probable hypotheses. Sorting is time consuming and may considerably slow recognition.

In [3] we proposed an *estimated rank pruning (ERP)* technique which combines the advantages of probability and rank pruning.

At each step  $i$  of the search ERP estimates the probability of the  $N_{\text{opt}}$ -th most likely hypothesis and prunes all less likely ones. The time complexity of ERP is similar to that of probability pruning and its search-space size, memory consumption, and recognition accuracy are comparable to those of rank pruning.

In this paper we outline two improvements of ERP. Using adaptive selection of the parameters used to determine the threshold, we reduce the average rank-estimation error by over 20% compared to results presented in [3]. We also address the potential danger of over-pruning where too many hypotheses are eliminated too soon in recognition process. If extreme over-pruning occurs, no valid end point may be active at the end of the recognition process, resulting in failure to produce any output. We show that ERP can be used to limit not just the maximum but also the minimum number of processed hypotheses, thereby reducing the chances for over-pruning. In our experiments limiting the minimum number of hypotheses enabled the decoder to run with very narrow pruning beam, which reduced significantly the running time (at the expense of loss in word error rate (WER)).

When compared with optimal values for lowest error rate, ERP used an average of 1.65% memory over the allowed maximum of the equivalent rank pruning algorithm, the set of hypotheses it maintained differed on average 5.54% from those maintained by rank pruning, recognition accuracy was the same and its running time was 35% shorter than that of probability pruning with the same accuracy and 48% shorter than that of exact rank pruning.

The ERP technique is described next. In Section 3 we describe the results obtained and compare them to rank and probability pruning. In Section 4 we present experiments with lower limit on the number of processed hypotheses in a frame and in 5 we briefly describe the system used to implement the algorithms.

## 2. ESTIMATED RANK PRUNING

Recall that an hypothesis is  $t$ -probable if its log probability is at most  $t$  below that of the most probable hypothesis at its step. Let  $N_i(t)$  be the number of  $t$ -probable hypotheses at step  $i$  and let  $t_i$  be the difference between the log probability of the most likely hypothesis at time  $i$  and that of the  $N_{\text{opt}}$ 'th most likely one at that step. Note that  $N_i(t_i) = N_{\text{opt}}$ .

Had we known  $t_i$ , we could efficiently mimic rank pruning by keeping only the  $t_i$ -probable hypotheses. But  $t_i$  is not known, and exact calculation of  $t_i$  by sorting would use the very operation we are trying to avoid. Instead, ERP uses a fast calculation to find a threshold  $\hat{t}_i$  such that  $N_i(\hat{t}_i) \approx N_{\text{opt}}$ .

To quickly determine  $\hat{t}_i$ , ERP uses an empirical observation indicating that  $N(t)$  is roughly exponential in  $t$ . Figure 1 shows a semi-log scale plot of  $N_i(t)$  as a function of  $t$  for several frames

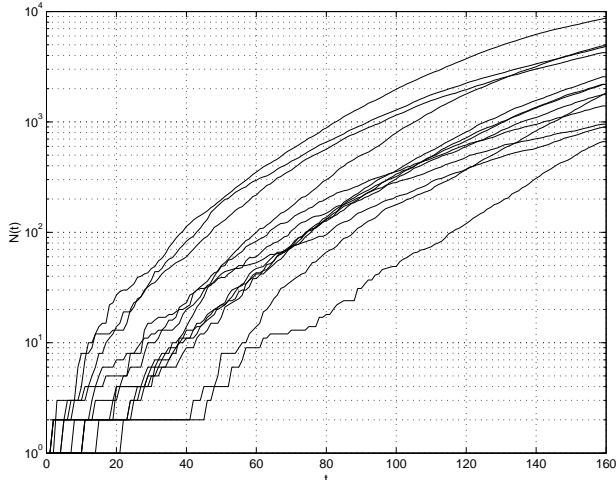


Fig. 1. Number of hypotheses as a function of the threshold

of a typical sentence. This relation appears in almost all sentences and frames. It suggests that over short threshold intervals,  $\log N_i(t)$  grows roughly linearly with  $t$ , namely,

$$N_i(t) \approx a_i \cdot e^{b_i \cdot t}, \quad (1)$$

for some constants  $a_i$  and  $b_i$ .

It follows that if we take two thresholds  $t'$  and  $t''$  near (the unknown)  $t_i$ , we can evaluate

$$b_i = \frac{\log N_i(t') - \log N_i(t'')}{t' - t''} \quad \text{and} \quad a_i = N_i(t') \cdot e^{-b_i t'}. \quad (2)$$

We then use Equation (1) to determine

$$\hat{t}_i = \frac{1}{b_i} \ln \frac{N_{\text{opt}}}{a_i}. \quad (3)$$

We then use  $\hat{t}_i$  as our estimate of  $t_i$  and maintain only the  $\hat{t}_i$ -probable hypotheses.

In our implementation we choose  $t' = t_{i-1}$  and  $t'' = (1 - \delta) \cdot t_{i-1}$  where  $\delta$  is a small constant, usually smaller than 0.1. Typically,  $t_i$  is not far from  $t_{i-1}$ , hence  $t'$  and  $t''$  are fairly close to  $t_i$ . Note that  $\hat{t}_{i-1}$  is not used to estimate  $\hat{t}_i$ , only to determine the points  $t'$  and  $t''$  on which the estimate of  $\hat{t}_i$  is based. Hence, if  $\hat{t}_i$  is very different from  $\hat{t}_{i-1}$ , the estimate will be less reliable, but not necessarily biased. The improvement that we will report in estimation accuracy over what was presented in [3] comes from the choice for  $\delta$ . Instead of using the fixed constant, we try to adopt it so that both  $N_i(t'')$  and  $N_i(t') - N_i(t'')$  are statistically significant. In other words if the number of  $t''$ -probable hypotheses is too small we decrease  $\delta$  and if  $N_i(t') - N_i(t'')$  is too small we increase  $\delta$ . The change in  $\delta$  means we need to recount again, but it happens rarely and it reduces the number of bad estimates significantly which is observable in the average error reduction.

The efficiency of the algorithm is increased by the pre-pruning step. All the non  $\hat{t}_{i-1}$ -probable hypotheses are pruned before the actual estimation of  $\hat{t}_i$ . The reason for this is to reduce the memory requirements of the algorithm and make it a better match to rank pruning in this aspect. It assumes that the previous threshold is close to the current one, and prunes all hypotheses that are not

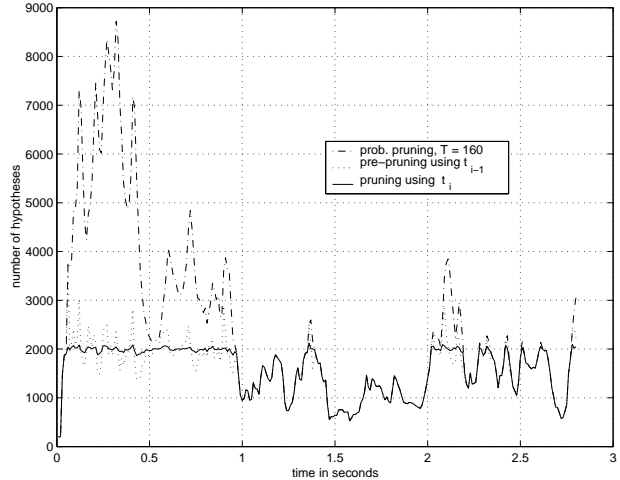


Fig. 2. Hypotheses selected by different thresholds,  $N_{\text{opt}} = 2000$

$\hat{t}_{i-1}$ -probable. However when  $\hat{t}_{i-1} < \hat{t}_i$  it may pre-prune some  $\hat{t}_i$ -probable hypotheses which should have been kept<sup>1</sup>.

The protocol can be summarized as follows:

At frame  $i$ :

1. Starting with the most likely and continuing in any order, extend all hypotheses keeping only  $\hat{t}_{i-1}$ -probable relative to the best extended hypothesis thus far.
2. Set  $t' = \hat{t}_{i-1}$  and  $t'' = (1 - \delta) \cdot \hat{t}_{i-1}$ .
3. Calculate  $N_i(t')$  and  $N_i(t'')$ .
4. Determine  $a_i$  and  $b_i$  using Equation (2).
5. Use Equation (3) to determine  $\hat{t}_i$ .
6. Set  $\hat{t}_i = \min(\hat{t}_i, T)$ .
7. Prune all but the  $\hat{t}_i$ -probable hypotheses.

Step 1 is the same as in other Viterbi searches. Steps 2, 4, 5, and 6 take a small constant time. Steps 3 counts hypotheses and is linear in their number. Step 7 eliminates all non  $t$ -probable hypotheses hence takes linear time as well; it is common to all search algorithms, and in practice is combined with step 1.

Figure 2 shows various pruning effects on the number of maintained hypotheses. The dashed, dotted, and solid lines show the number of hypotheses maintained by probability pruning algorithm with threshold  $T = 160$ , after pre-pruning step and by ERP if there were no pre-pruning, respectively. Whenever the dotted line falls below the solid line, the pre-pruning step eliminates hypotheses that should have been preserved.

### 3. PERFORMANCE EVALUATION

In the first set of experiments we wanted to compare the performance of the three pruning algorithms: probability, rank and ERP. Figure 3 shows the running times relative to actual speech duration and the corresponding WER for various parameter settings in the three algorithms. In probability pruning the threshold was changed

<sup>1</sup>Some of these hypotheses may still survive as the probability of the best hypothesis extended before them may be lower than that of the best extended hypothesis at the end.

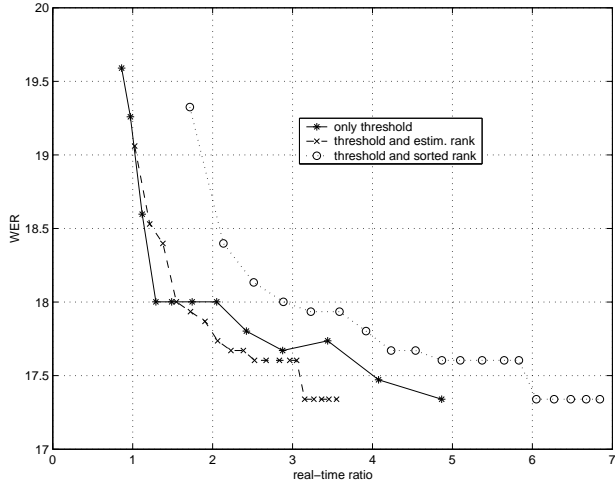


Fig. 3. Running time vs. accuracy

from 95 to 150 in the steps of 5. For higher values of 150 we don't witness any change in recognition accuracy. Both rank and ERP procedures had threshold fixed at  $T = 150$ , and maximum number of hypotheses was changing as in Table 1.

We can observe that ERP is performing better than the other two techniques. For example, for  $N_{\max} = 4000$  both rank pruning methods achieve the best possible WER, and ERP is 48% faster than exact rank and about 35% faster than probability pruning with  $T = 150$ . From Figure 4 we can observe that estimated and exact rank pruning methods have essentially the same WER for the same target  $N_{\max}$ .

Worth noting is that our system uses simple monophone acoustic models as will be described in Section 5, and time consumed by sorting operation dominates the recognition. In large vocabulary systems models are more complicated and sorting takes smaller fraction of running time. That means that although expected savings with ERP compared to exact rank pruning in absolute time should be higher (usually  $N_{\max} = 10000$ ), the relative savings would not be as high as in this system. This also explains why rank pruning does not improve run-time performance of this system compared to simple probability pruning as it is reported for large vocabulary systems.

Next we evaluate the memory usage and rank estimation accuracy of the ERP algorithm. Let  $M_i$  denote the number of hypotheses maintained by the ERP algorithms at frame  $i$ , just after Step 1, and define  $miss = \frac{|N_i(\hat{t}_i) - N_{opt}|}{N_{opt}}$ , and  $over = \max\left(0, \frac{M_i - N_{opt}}{N_{opt}}\right)$ . When computing their average values,  $miss$  is averaged over all frames when  $t_i < T$ , namely where  $N_i(T) > N_{opt}$ , and  $over$  is averaged over all frames since we may have  $M_i \geq N_{opt}$  even when  $N_i(T) < N_{opt}$  (due to the uncertainty of the best score during pre-pruning step).

Average values for  $over$  suggest that the method has successfully addressed the memory limitations. The fact that peak values are relatively high is not alarming since the average values show that it can not be frequent. In designing the system that uses ERP we should still be prepared to handle more hypotheses than  $N_{\max}$ . The error for rank miss is also very small, and peak values are not very disturbing. Important observation is that the error comes from

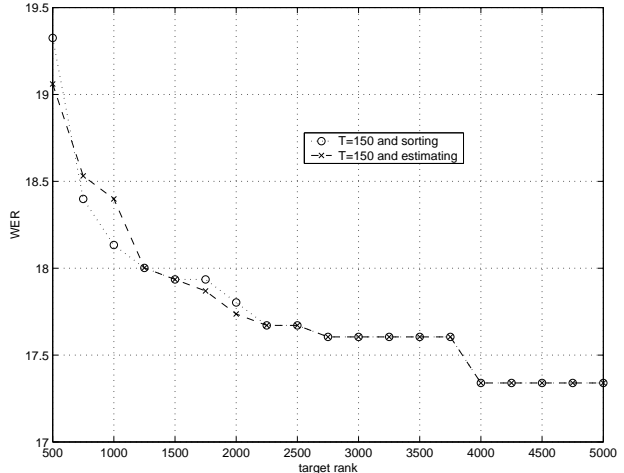


Fig. 4. WER comparison for estimated and exact rank pruning

$N_{\max}$	xRT rank pruning		rank miss		over	
	estim.	exact	average	maximum	average	maximum
500	1.026	1.716	7.59%	434.8%	7.60%	434.8%
750	1.214	2.135	7.16%	90.1%	6.44%	231.3%
1000	1.378	2.514	6.95%	68.2%	5.70%	236.7%
1250	1.547	2.885	6.81%	68.6%	5.08%	177.7%
1500	1.723	3.228	6.68%	66.1%	4.53%	174.7%
1750	1.905	3.587	6.53%	64.9%	4.08%	178.3%
2000	2.063	3.919	6.44%	64.5%	3.66%	162.1%
2250	2.229	4.232	6.27%	65.4%	3.30%	168.0%
2500	2.384	4.539	6.17%	64.2%	2.98%	180.3%
2750	2.525	4.867	6.02%	64.1%	2.70%	159.6%
3000	2.669	5.098	5.91%	62.7%	2.44%	137.9%
3250	2.835	5.371	5.79%	62.6%	2.20%	119.6%
3500	2.965	5.647	5.70%	62.3%	2.00%	103.9%
4000	3.152	6.050	5.54%	60.1%	1.65%	82.4%
4500	3.364	6.479	5.37%	59.5%	1.37%	79.3%
5000	3.550	6.843	5.23%	59.3%	1.15%	73.4%

Table 1. Performance comparison with Exact rank pruning

two sources. One is pre-pruning and the other is bad estimation. In the next Section we will show that dominant source of error is pre-pruning stage and that the estimation model is on average very accurate.

#### 4. LIMITING MINIMAL NUMBER OF HYPOTHESES

At some stages of the recognition process the number of  $T$ -probable hypotheses may be very small. This may lower the system's robustness if, as sometimes happens, an incorrect hypothesis scores highly. For example, we have observed cases where the recognizer reached the end of the utterance with no valid final node to end the recognition in, and was thus unable to output a result.

To remedy the situation, we propose to always keep at least a prescribed number  $N_{\min}$  of hypotheses. This can be accomplished easily using a slight modification of ERP, which we assume is already implemented to limit the maximum number of hypotheses at any frame.

As in the previous section, the algorithm uses the threshold  $T$  for pruning whenever  $N_{\min} \leq N_i(T) \leq N_{\max}$  and if  $N_i(T) > N_{\max}$  it estimates a threshold  $\hat{t}_i < T$  such that  $N_i(\hat{t}_i) = N_{\max}$ . However, if  $N_i(T) < N_{\min}$ , it finds a threshold  $\hat{t}_i > T$  that

satisfies  $N_i(\hat{t}_i) = N_{\min}$ . Consequently Step 6 in the algorithm is changed to

$$\hat{t}_i = \max(\hat{t}(N_{\min}), \min(T, \hat{t}(N_{\max}))).$$

One error source in limiting the maximum number of hypotheses was the pre-pruning stage. When trying to ensure  $N_i(\hat{t}_i) \geq N_{\min}$ , we do not have to worry about exceeding the physical memory available to the system as  $N_{\min}$  is typically much smaller than  $N_{\max}$ . Let us define  $\hat{t}_i^+$  to be a solution of  $N_i(\hat{t}_i^+) = (1 + \gamma)N_{\min}$ . We modify the pre-pruning threshold in Step 1 of the algorithm to  $\max(\hat{t}_{i-1}, \hat{t}_{i-1}^+)$ . This effectively targets at least  $(1 + \gamma)N_{\min}$  hypotheses for the next frame. To ensure that the *over* parameter does not increase, we need only guarantee that  $(1 + \gamma)N_{\min} \leq N_{\max}$ . In our experiments we used  $\gamma = 0.25$ . As an additional safety procedure, after Step 6, when the threshold has already been determined, if the number of hypotheses that survived the pre-pruning is lower than  $0.9 \cdot N_{\min}$  we repeat the frame, this time keeping all  $\hat{t}_i$ -probable hypotheses.

Two observations are worth mentioning. First, repeating the frame is not as expensive as running it for the first time as all observable parameters have already been matched to relevant phone models and cached values are still available for recombination. For  $\gamma = 0.25$ , the typical number of repeated frames is 4.5% which increases the running time of recognition by less than 1%. Second, repeating the frame does not affect our threshold estimation, only allows maintaining all  $\hat{t}_i$ -probable hypotheses. If  $\hat{t}_i$  is a bad estimate of  $t_i$ ,  $N(\hat{t}_i)$  may be very different from  $N_{\min}$ .

Our language model is not very restrictive as it accepts any sequence of dictionary words, and any word end context is a valid recognition end point. However, for some sentences in our test set, and for thresholds lower than 95, the system does not generate recognition output because all valid end word nodes have been pruned. The fastest running time achievable with probability pruning at  $T = 95$  is 0.862 relative to real-time. The corresponding WER is 19.59%. Table 2 shows experiments in which  $T = 80$  while varying  $N_{\min}$ . Observe that it was possible to run the recognizer 19.26% faster than with  $T = 95$  but with a higher WER. Additionally, the WER decreases as  $N_{\min}$  increases, but at a smaller payoff for increase in running time compared to the payoff from conventional threshold variation. Notice that errors in rank estimation for  $N_{\min}$  are much smaller than for  $N_{\max}$ . The difference is due to the pre-pruning effects which dominate the error in the  $N_{\max}$  estimation. As in Table 1, the errors in rank estimation increase with reducing the target number of hypotheses as the collected statistics ( $N(t')$  and  $N(t'')$ ) are less statistically reliable.

We conclude that the technique helps prevent over-pruning as recognition was possible with a tighter beam. Another conclusion is that the technique can only be used for a small  $N_{\min}$ , to prevent over-pruning, and not for adjusting the recognition performance, because the payoff in WER for increasing recognition time is higher when adjusting threshold. A third conclusion is that the threshold estimation technique is very accurate as the average errors in estimating  $\hat{t}(N_{\min})$  were lower than 2%.

## 5. THE SYSTEM

The evaluation tests were performed on the TIMIT database using a Java-based speech-recognition system that is being built at UCSD. We note however that the test was somewhat unfair as the recognizer used pronunciation and language models based partly on the test set. Therefore the results described reflect the relative

$T = 80, N_{\max} = 1500$						
$N_{\min}$	xRT	WER	average miss		maximum miss	
			$N_{\max}$	$N_{\min}$	$N_{\max}$	$N_{\min}$
80	0.696	23.7%	5.74%	4.44%	28.7%	1375.0%
100	0.733	23.3%	6.26%	3.92%	28.7%	387.0%
150	0.799	22.8%	5.83%	3.20%	28.7%	386.7%
200	0.850	23.0%	5.47%	2.91%	28.7%	501.5%
350	1.024	21.7%	5.29%	2.28%	28.7%	79.7%
500	1.176	19.3%	5.10%	2.01%	28.7%	100.6%
650	1.332	18.8%	6.57%	1.87%	85.3%	81.1%
800	1.474	18.7%	6.02%	1.78%	95.9%	84.3%
950	1.645	18.9%	6.69%	1.69%	95.9%	82.0%
1100	1.823	18.3%	6.45%	1.65%	99.1%	93.0%

Table 2. Performance when limiting  $N_{\min}$

merits of the various pruning techniques and should not be used for comparison with other baseline systems.

The front end used 39-dimensional feature vectors, 12 mel-based cepstral coefficients, energy, and their first and second derivatives obtained through linear regression [4]. Frame width was 25ms and frame shift was 10ms. The system used 48 speaker-independent monophones as defined in [5]. They were modeled as 3-state, left-to-right, HMMs with at most ten Gaussian mixture components. Short pause was modeled as a single state HMM. The models were trained on the training portion of TIMIT. The tests were performed on TIMIT core test set.

The language model used bigrams based on both the training and testing sets. Its back-off probabilities were determined by a Good-Turing estimate. The total number of words in TIMIT is 6100, but due to the size of the data only about 950 contexts were created. The average perplexity of the core test set was 1741.

The pronunciations dictionary used all pronunciations appearing in both the training and testing sets according to the hand-labeled transcriptions. Each pronunciation was assigned a probability proportional to its frequency. The final network was obtained by composition of the language model network and the pronunciation dictionary. The total number of vertices in final network was 112138 and the number of edges was 184450. Hypotheses corresponded to edges.

The test was performed on a Pentium 3 - 500MHz computer with 128MB of memory.

## 6. REFERENCES

- [1] H. Ney and S. Ortman. Dynamic programming search for continuous speech recognition. *IEEE Signal Processing Magazine*, 16(5):64–83, Sept. 1999.
- [2] N. Desmukh, A. Ganapathiraju, and J. Picone. Hierarchical search for large-vocabulary conversational speech recognition: working toward a solution to the decoding problem. *IEEE Signal Processing Magazine*, 16(5):84–107, Sept. 1999.
- [3] N. Jevtić, A. Klautau, and A. Orlitsky. Estimated rank pruning and java-based speech recognition. *Automatic Speech Recognition and Understanding Workshop*, December 2001.
- [4] J. Picone. Signal modeling techniques in speech recognition. *Proceedings of the IEEE*, 81(9):1215–47, Sep. 1993.
- [5] K.-F. Lee and H.-W. Hon. Speaker-independent phone recognition using hidden markov models. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 37(11):1641–8, Nov. 1989.