

Improved Modeling and Efficiency for Automatic Transcription of Broadcast News

Ananth Sankar¹ Venkata Ramana Rao Gadde
Andreas Stolcke Fuliang Weng²
Speech Technology and Research Laboratory
SRI International, Menlo Park, CA, U.S.A.

To appear in *Speech Communication*

¹Now with Nuance Communications, Menlo Park, CA 94025, USA

²Now with Intel Corporation, Beijing, China

Abstract

Over the last few years, the DARPA-sponsored Hub-4 continuous speech recognition evaluations have advanced speech recognition technology for automatic transcription of broadcast news. In this paper, we report on our research and progress in this domain, with an emphasis on efficient modeling with significantly fewer parameters for faster and more accurate recognition. In the acoustic modeling area, this was achieved through new parameter tying, Gaussian clustering, and mixture weight thresholding schemes. The effectiveness of acoustic adaptation is greatly increased through unsupervised clustering of test data. In language modeling, we explored the use of non-broadcast-news training data, as well as adaptation to topic and speaking styles. We developed an effective and efficient parameter pruning technique for backoff language models that allowed us to cope with ever increasing amounts of training data and expanded N-gram scopes. Finally, we improved our progressive search architecture with more efficient algorithms for lattice generation, compaction, and incorporation of higher-order language models.

Zusammenfassung

In jüngster Zeit wurde die automatische Transkription von Rundfunknachrichten durch die von der amerikanischen DARPA geförderten Hub-4-Spracherkennungswettbewerbe vorangetrieben. In diesem Artikel berichten wir über Fortschritte auf diesem Gebiet, mit einem Schwerpunkt auf effizienter Modellierung mit weniger Parametern zwecks beschleunigter und genauerer Spracherkennung. In der akustischen Modellierung wurde dies erreicht durch neue Verfahren zur Parameterbindung, zum Clustern von gaußschen Verteilungen und zum Komprimieren von Gewichten in Mischverteilungen. Die Effektivität von akustischer Adaptierung wurde durch automatisches Clustern der Testdaten erheblich verbessert. In der Sprachmodellierung untersuchten wir die Benutzung von Trainingsdaten außerhalb der Rundfunknachrichten-Domäne sowie die Anpassung an Themen und Sprechstil. Wir haben ein effektives und effizientes Verfahren zum Parameter-Pruning in Backoff-Sprachmodellen entwickelt, das es uns ermöglicht, stetig wachsende Trainingskorpora und längere N-gramme zu verwenden. Abschließend beschreiben wir Verbesserungen in der progressiven Sucharchitektur unseres Erkenners, mit effizienteren Algorithmen zur Erzeugung, Komprimierung und Expandierung von Wortgraphen.

Résumé

Durant ces dernières années, les évaluations Hub-4 des systèmes de reconnaissance de la parole continue sponsorisées par DARPA ont fait progresser les techniques de reconnaissance de la parole pour la transcription de nouvelles audio-diffusées (“broadcast news”). Dans cet article, nous présentons notre recherche et nos progrès dans ce domaine, en nous concentrant plus particulièrement sur une modélisation efficace utilisant sensiblement moins de paramètres, permettant ainsi d’améliorer la vitesse et les performances de la reconnaissance vocale. En termes de modélisation acoustique, les améliorations ont été obtenues en utilisant une nouvelle méthode pour l’ajustement des paramètres, l’agrégation des Gaussiennes, et le seuillage des poids des mélanges de Gaussiennes. L’efficacité de l’adaptation acoustique est grandement améliorée par l’agrégation non supervisée des données de test. En modélisation du langage, nous avons étudié le résultat de l’utilisation de données d’entraînement ne provenant pas de “broadcast news”, ainsi que de l’effet de l’adaptation au sujet et au style de parole. Nous avons développé une technique efficace d’élagage des paramètres pour des modèles de langage avec repli (“backoff”) ce qui nous a permis de supporter l’accroissement continu de la quantité de données d’entraînement et l’extension de la portée des N-grammes. Enfin, nous avons amélioré notre architecture de recherche progressive en utilisant des algorithmes plus efficaces pour la génération et la compaction de treillis, ainsi qu’en y incorporant des modèles de langage d’ordre supérieur.

1 Introduction

In recent years there has been increasing interest in developing large-vocabulary continuous speech recognition (LVCSR) systems for speech found in real sources. Broadcast news, in particular, has been the testbed for the DARPA-sponsored Hub-4 continuous speech recognition (CSR) evaluations over the last few years, and represents a significant challenge to speech recognition researchers.

Many interesting problems are associated with the automatic recognition of broadcast news. One problem is that the speech is in the form of a single long stream, whereas typical automatic speech recognition (ASR) systems are designed to process sentence-length units of speech. ASR systems work best when the segment to be recognized is homogeneous with respect to speaker and acoustic condition. It is also desirable, both for ASR and for speech understanding, that the segments correspond to linguistic units such as sentences or phrases. An interesting challenge, therefore, is to develop algorithms that can automatically segment a long stream of speech according to such criteria. Another problem with broadcast news is the many different variations of speech, such as conversational speech, noisy speech, speech in the presence of music, non-native speech, or a combination of these variations. It is necessary to develop techniques that are robust to these variations. Finally, it is important to focus attention on real-time recognition to transfer this technology to applications such as information archival and retrieval.

In this paper, we review our work on broadcast news transcription over the last three years. The paper is divided into sections detailing work in acoustic processing and modeling, language modeling, and lattice-based decoding. Comparative experimental results showing the performance of the techniques are given in each section.¹

2 Broadcast News Task Description

The data used for the Hub-4 evaluation task consists of recordings of broadcast news shows from various television and radio sources. In contrast to previ-

¹However, since this work was done over the course of the last few years, the reader should be careful to note that the same test sets and baseline systems are not used across different sections.

ous test sets used in DARPA evaluations, the Hub-4 data is naturally occurring data, collected under realistic conditions. It is a mixture of various different speech styles, acoustic conditions, and background non-speech segments, making the recognition task difficult. The Hub-4 data has been divided into seven different acoustic focus conditions to facilitate the study of different speech recognition problems in this data. These conditions are:

- F0** : Clean read speech (e.g., broadcast news anchor)
- F1** : Conversational speech (e.g., interviews in the news studio)
- F2** : Telephone speech (e.g., telephone interviews)
- F3** : Speech with background music (e.g., introduction of stories)
- F4** : Noisy speech (e.g., field interviews; noisy channels)
- F5** : Non-native speech (e.g., non-native reporters)
- FX** : Anything that could not be classified into the previous categories

Two types of Hub-4 test data have been used by the community: (1) the partitioned evaluation (PE) data and (2) the unpartitioned evaluation (UE) data. The PE data consists of manually-created acoustically homogeneous segments. The UE data, on the other hand, is simply the original recording from the broadcast news show. All recent DARPA evaluations have used only UE test data. A more detailed description of the Hub-4 data is given in Stern (1997).

Until the 1998 Hub-4 evaluation, the basic evaluation metric was the system word error rate. In 1998, a 10 times real-time spoke evaluation was introduced to focus attention on real applications.

3 Acoustic Processing and Modeling

3.1 Acoustic Segmentation and Clustering of UE data

Processing a single long UE segment containing both speech and significantly long non-speech segments is difficult. Thus, we first chop the UE data into segments of manageable length that contain only speech. For segmentation, we first run a fast recognition

step on the UE segment using a parallel male/female context-dependent (CD) phonetically tied mixture (PTM) hidden Markov model (HMM) set to produce a hypothesized sequence of gender-tagged words and background segments (Sankar, Weng, Rivlin, Stolcke, and Gadde, 1998). Both the words and background segments are time-tagged. The models used for this recognition step are trained using the Hub-4 training data, and 5 minutes of non-speech segments (silence, background noise and music) are used to train the background model.

The segmentation algorithm processes the output from the recognition step. It first removes any non-speech segments longer than one second, and then chops at the remaining non-speech regions to create nominally 10-second segments. In addition, a new segment is created whenever a gender change occurs. The resulting segments are thus nominally 10 seconds long, and are labeled by gender. This algorithm does not attempt to make sure that the resulting segments are acoustically homogeneous. A segment may contain multiple speakers or speech from different acoustic focus conditions.

To facilitate adaptation to the test data, the segments were clustered using bottom-up agglomerative clustering (Sankar et al., 1998; Sankar, Heck, and Stolcke, 1997; Heck and Sankar, 1997). The distance measure used for clustering is derived as follows: First we train a Gaussian mixture model (GMM) using all the test segments, and a separate mixture weight distribution for each segment to these shared Gaussians. The mixture weight distribution for a cluster of segments is simply the weighted-by-counts average of the individual distributions. The distance between two segment clusters is then defined as the weighted-by-counts increase in entropy of the mixture weight distribution due to merging the two clusters (Sankar et al., 1998). This is similar to the approach we use for HMM state clustering (Digalakis, Monaco, and Murveit, 1996a).

We tested the segmentation algorithm with the 1996 Hub-4 development test data. For four of the test shows in this data, manually created PE segments were available. We could thus compare our automatically created segments against these. We ran recognition on the manually created and automatically created PE segments for these shows using a 20,000-word bigram language model (LM), and non-crossword gender-

PE Segment Type	Models	
	SI	Cluster-Adapted
Manually created	37.9	35.5
Automatically created	39.4	37.6

Table 1: Word error rates (%) for the PE and UE segments

dependent HMMs. Both speaker-independent (SI) models and models adapted to the segment clusters using maximum likelihood (ML) transformation-based adaptation (Sankar and Lee, 1994, 1996; Digalakis, Rtischev, and Neumeyer, 1995; Legetter and Woodland, 1995) were used. Table 1 gives word error rates for the manually created and automatically created segments. In the case of the SI models, the word error rate was 1.5% (absolute) worse for the automatically created segments. However, for the adapted models this difference increased to 2%. This is probably because our segmentation algorithm does not guarantee acoustically homogeneous segments, thus not deriving maximum benefit from acoustic adaptation.

3.2 Modeling the Acoustic Focus Conditions

SRI’s acoustic modeling technology is based on state-clustered HMMs. Acoustically similar HMM states are clustered together, each cluster sharing a set of Gaussian distributions. Each HMM state in a cluster has a different set of mixture weights associated with the shared Gaussians (Digalakis et al., 1996a). In SRI’s system, the shared Gaussians are called “Genones”, and the models “Genonic HMMs”. The observation density for state i is given by

$$p_i(x) = \sum_{m=1}^M w_{i,m} N_{g,m}(x), \quad (1)$$

where state i belongs to state cluster g , $N_{g,m}$ is the m th Gaussian distribution corresponding to state cluster g , M is the number of shared Gaussians, and $w_{i,m}$ is state i ’s mixture weight for the m th Gaussian. The observation density parameters and other HMM parameters

are estimated using ML training. The expectation-maximization (EM) algorithm (Dempster, Laird, and Rubin, 1977; Juang, 1985) is used to iteratively increase the likelihood of the models using transcribed training data.

Since the acoustic focus conditions in the Hub-4 data are so different from each other, we decided, in our initial work, to train a separate gender-specific Genonic HMM for each focus condition (Sankar et al., 1997). However, the first release of the Hub-4 training data contained only 50 hours of data, 35 hours of which were speech segments and the rest non-speech segments not usable for training. This data accounted for all the conditions; thus there was not enough data to train good condition-specific models. We addressed this problem by using ML transformation-based adaptation (Sankar and Lee, 1994, 1996; Digalakis et al., 1995; Legetter and Woodland, 1995). We adapted a model trained on the Wall Street Journal (WSJ) database (Doddington, 1992) to each of the focus conditions except for the F1 condition. For the F1 condition, we used the Switchboard corpus (Godfrey, Holliman, and McDaniel, 1992) to train the seed models. We did this because both Switchboard and F1 contain conversational speech. While the idea of condition-specific models made intuitive sense, work done by BBN (Kubala et al., 1997) showed that a single model trained on all the Hub-4 data performed better. Motivated by their results, we also compared the performance of our condition-specific models to a single gender-specific Hub-4 model trained on the 50 hours of Hub-4 training data (Sankar et al., 1998).

Table 2 gives recognition word error rates for the male subset of the 1996 Hub-4 development test set. Recognition was run from bigram lattices (Murveit, Butzberger, Digalakis, and Weintraub, 1993) generated using the 20,000 word bigram language model (LM) we used for the 1996 evaluations. The single Hub-4 model gave a relative 6.1% lower word error rate than the condition-specific models. Since training a single Hub-4 model is easier, we have since been using this approach to train broadcast news models.

It is possible that with a large amount of training data, it would be possible to train good condition-specific models. Currently there are 200 hours of Hub-4 training data available; however, we have not repeated condition-specific model experiments with this data. One disadvantage of condition-specific models

Models	Word Error (%)
Condition-Specific	41.12
Single Hub-4 Model	38.61

Table 2: Comparison of condition-specific models vs. a single Hub-4 model

is that it is necessary to automatically determine the acoustic condition of the test data, which is not an easy task in itself. Most broadcast news systems currently use one or at most two, models. In the latter case, a separate model is used for broadband speech and for telephone speech (for example, see Woodland et al. (1998)).

3.3 New Training Algorithms for Acoustic Models

The EM algorithm is an iterative algorithm that recomputes the model parameters, based on their current values, so as to increase the model likelihood with each iteration (Dempster et al., 1977). The algorithm performance is dependent on the initial parameter values, and can, at best, arrive at a locally optimal solution. To investigate this issue, we developed and studied several techniques for HMM parameter initialization and training (Sankar, 1998a).

Another important problem in training acoustic models for speech recognition is that of robustly estimating a large number of parameters with limited data. We addressed this problem by developing new HMM training algorithms based on previously developed acoustic adaptation techniques (Sankar, 1998c).

3.3.1 Parameter Initialization and Training

3.3.1.1 SRI's Previous Training Algorithm. We start with a brief description of SRI's previous HMM training algorithm. By way of example, consider the problem of training an HMM with 1000 state clusters and 32 Gaussians in each of the corresponding Genones. This is done by first training a PTM system, where all states in a phone share the same set of 100 Gaussians. The states in this phone are then clustered

using bottom-up agglomerative clustering (Digalakis et al., 1996a), and a cut in the cluster tree is chosen so as to give 1000 state clusters.

The 32 Gaussians in each state cluster are initialized using the corresponding 100 PTM Gaussians. The 100 Gaussians in each phone are clustered down to 32 for each state cluster through a series of steps involving the selection of the most likely Gaussians for each state cluster, and also Gaussian merging. Details of the algorithm can be found in Digalakis et al. (1996a).

This approach poses the following potential problem for the initial values of the Gaussians in the state clusters and hence the final models: The 100 PTM Gaussians cover the entire acoustic space for a particular phone; however, each state cluster for this phone covers only a small part of this large acoustic space. Thus, the PTM Gaussians may not be appropriate for initializing the Gaussians in the individual state clusters, and may result in inefficient use of the parameters.

3.3.1.2 Gaussian Splitting and Merging. We implemented a Gaussian initialization scheme based on the splitting strategy commonly used in vector quantization (Linde, Buzo, and Gray, 1980; Gersho and Gray, 1991). In this approach, we first estimate a single Gaussian model for each Genone. We then split the Gaussian for each Genone into two by slightly perturbing the mean of the Gaussian along the direction of the standard-deviation vector, and reestimate the model by further EM training. This process of splitting and retraining is repeated until the required number of Gaussians is achieved. At each stage, we can choose how many Gaussians to split. Thus, if there are currently n Gaussians which we want to increase to m Gaussians, then we split the $m - n$ Gaussians which have the largest average sample variance (Sankar, 1998a). A similar Gaussian splitting algorithm is used in the Cambridge University HTK system, though a different criterion is used to select which Gaussian to split (Young and Woodland, 1993).

The Gaussian splitting approach can be configured in a variety of ways. For example, we may split all Gaussians at each stage, or may split only the single largest variance Gaussian, or may do something in between these extremes. We experimented with many of these approaches and found that there was not a very significant difference in performance. Thus, we

decided on a simple strategy that splits all Gaussians at each stage until we have the desired number of Gaussians per Genone.

The Gaussian splitting algorithm tends to uniformly distribute the training data among the Gaussians (Sankar, 1998a). Thus, if a Genone has very little data, then all its Gaussians may be poorly estimated. To ensure robust Gaussian estimation, we used a Gaussian merging algorithm. In this method, the Gaussians in a Genone are iteratively merged using bottom-up agglomerative clustering until all Gaussians have at least a threshold amount of data (Sankar, 1998a). This threshold is specified by the user, and its optimum value is experimentally determined. For clustering, the distance between two Gaussians is given by the weighted-by-counts increase in entropy due to merging the Gaussians. Combining Gaussian merging and splitting by doing a merge operation before every split operation gives the GMS algorithm (Sankar, 1998a).

3.3.1.3 Experimental Results. For these experiments, we used the Wall Street Journal (WSJ) corpus (Doddington, 1992). We trained HMMs using a small subset of the WSJ SI-284 male training data. We used 71 of the 142 male training speakers and about 50 sentences from each for a total of about 3500 training sentences. We created three different WSJ test sets, denoted as WSJ1, WSJ2, and WSJ3, each with 10 male speakers and about 3600 words, for a total of about 10,900 words in all. For speed of experimentation, recognition was run from bigram lattices (Murveit et al., 1993).

We trained two sets of acoustic models: the first had 991 state clusters and the second had 2027. Both models had 32 Gaussians per Genone. We trained the models using both our previous training algorithm and the GMS algorithm. Table 3 shows that the GMS algorithm performs similarly to the old method for the 991 Genone model, but is significantly better for the 2027 Genone model, where the number of parameters is very large relative to the amount of training data. This shows the robustness of the GMS algorithm relative to our previous approach.

Since speaker adaptation (Sankar and Lee, 1994, 1996; Digalakis et al., 1995; Legetter and Woodland, 1995) is a common technique used in most state-of-the-art systems, we studied the interaction between the training algorithm for the original speaker-

Database	Word Error Rate (%)			
	Old algorithm		GMS algorithm	
	Number of Genones			
	991	2027	991	2027
WSJ1	23.7	25.3	23.5	23.9
WSJ2	13.7	15.5	13.5	14.1
WSJ3	24.3	26.0	23.9	25.1

Table 3: Comparison of word error rates (%) for systems with different numbers of parameters

Models	Training algorithm for SI models			
	Old		GMS	
	Number of Genones			
	991	2027	991	2027
Speaker-independent	23.7	25.3	23.5	23.9
Adapted	20.5	21.1	19.9	20.3

Table 4: Comparison of word error rates for WSJ1 before and after adaptation using different approaches to train the SI models

independent (SI) models and speaker adaptation. Table 4, shows the word error rate before and after speaker adaptation for WSJ1 for the 991 and 2027 Genone systems when the SI models were trained with the old algorithm and the GMS algorithm. The results show that the performance both before and after adaptation was superior with the GMS algorithm.

3.3.2 Robust Parameter Estimation

While the GMS algorithm gives robust parameter estimates, it does so indirectly by merging Gaussians that have too little data. Thus, if there is less training data, fewer Gaussians will be trained, giving less acoustic resolution. To train a large number of Gaussians, we developed the tied-transform HMM (T^2 -HMM), which uses ML acoustic adaptation methods to robustly estimate HMMs with a large number of parameters (Sankar, 1998c).

3.3.2.1 Tied-Transform HMM. We explain the concept of T^2 -HMMs using the state-cluster tree in Figure 1. The leaf nodes in the tree correspond to individual HMM states. The other nodes in the tree correspond to state clusters that are created using bottom-up agglomerative clustering (Digalakis et al., 1996a). State clusters can be generated by cutting the tree at intermediate levels. The figure shows state clusters at two different levels with N and M state clusters, where $N > M$.

Suppose our goal is to train an HMM for the larger number of state clusters N . However, we do not have enough data to robustly estimate the large number of Gaussians in this system. In the T^2 -HMM, we solve this problem by training an HMM for the smaller number of state clusters M , for which we assume we have enough data to robustly estimate each Gaussian. We can always select a small enough M so that robust Gaussian estimates are possible. Each state cluster in the larger HMM is a descendent of an ancestor state cluster in the smaller HMM as shown in the figure. The Gaussians in the state clusters of the larger HMM are transformed versions of the ancestor Gaussians in the smaller HMM. In the figure, the transformations $T(1), \dots, T(m)$ are used to map the Gaussians in the Genone $GEN(0)$ to the Gaussians in the Genones $GEN(1), \dots, GEN(m)$. $T(i)$ can also be a set of transforms, each tied to a cluster of acoustically similar Gaussians in the Genone $GEN(i)$. Since the transforms are tied to a set of Gaussians in the N -state-cluster HMM, they can be estimated with the pooled data from all those Gaussians. This results in robust estimates of the transforms.

The estimation problem is now that of computing the parameters of the smaller HMM and the parameters of the transformations. We can use different types of transformations as have been described in the acoustic adaptation literature (Sankar and Lee, 1994, 1996; Digalakis et al., 1995; Legetter and Woodland, 1995; Neumeyer, Sankar, and Digalakis, 1995). In this paper, we chose to use the block-diagonal affine matrix transform of the Gaussian means as this has given us good performance in the past for speaker adaptation (Neumeyer et al., 1995). We solve the ML estimation problem iteratively. First, we assume identity transforms and estimate the parameters of the smaller HMM. Then we keep the parameters of the small HMM fixed, and estimate the transformations.

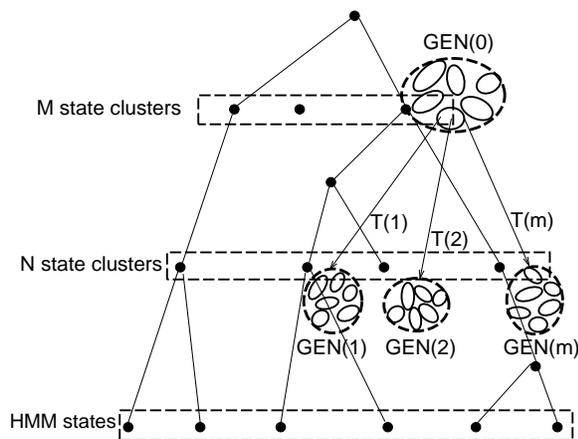


Fig. 1: Illustration of T^2 -HMM

This procedure can be iterated. However, in our experiments, we used only one iteration of this approach. The ML estimation of HMM parameters is well established, and that of the transformations has previously been studied in the context of acoustic adaptation (Sankar and Lee, 1994, 1996; Digalakis et al., 1995; Legetter and Woodland, 1995).

The T^2 -HMM idea is related to that of maximum a posteriori (MAP) estimation of HMM parameters (Gauvain and Lee, 1994). In MAP estimation too, a small HMM can be adapted to a large HMM, but using MAP smoothing, instead of ML transformation-based adaptation as in T^2 -HMMs. The T^2 -HMM approach has the advantage that we need to store only the parameters of the small HMM and the tied transformation parameters, while in the MAP approach, all the Gaussian parameters of the large HMM must be individually stored. Depending on the number of transforms used, this can result in a significant reduction in storage for the T^2 -HMMs.

3.3.2.2 Experimental Results. For training we used the male subset of the 100 hours of Hub-4 training data released by NIST for the 1997 DARPA-sponsored Hub-4 evaluation. For testing, we used the 1996 Hub-4 male development test set. We ran recognition using trigram lattices generated with the algorithms described in Section 5.

Table 5 gives the recognition word error rates on this test set comparing the GMS algorithm, the T^2 -

	GMS	MAP smoothing	T^2 -HMM
Number of clusters			
	2209	8409	8409
32 Gaussians per cluster			
Number of Gaussian parameters in Millions			
	5.5	30	9.8
F0	14.2	15.6	14.2
F1	30.5	30.7	29.3
F2	37.5	38.2	36.2
F3	29.0	30.8	30.5
F4	27.5	27.4	26.2
F5	28.2	29.3	28.0
FX	56.4	56.2	56.0
All	31.4	32.0	30.7

Table 5: Comparison of word error rates (%) for different training algorithms on the 1996 Hub-4 development data

HMM approach, and a MAP smoothing approach similar to that of Gauvain and Lee (1994). We trained a crossword state clustered HMM with 2209 state clusters, and one with 8409 clusters. The 2209-cluster system is the one we used for the 1997 Hub-4 evaluations (Sankar et al., 1998). Table 5 shows that the 8409-cluster model gives worse performance than the 2209-cluster model when trained using the GMS algorithm. However, both the T^2 -HMM and MAP smoothing algorithms gave an improvement over the GMS algorithm for the 8409-cluster system (32.0% to 30.7%). A smaller, but significant, improvement is observed over the 2209-cluster system (31.4% to 30.7%).

From Table 5, we see that the MAP smoothing algorithm and the T^2 -HMM algorithm gave the same word error rate. However, the T^2 -HMM can be stored much more efficiently, because we need to store only the smaller HMM and the set of transforms, as opposed to the MAP algorithm, where we must independently store each Gaussian in the larger model. In particular, as shown in Table 5, the T^2 -HMM needs a factor of three fewer parameters to store the Gaussian distributions as compared to the MAP-trained HMM.

3.4 Acoustic Adaptation

There is often a mismatch between the training and test data, leading to degraded performance on the test data. Acoustic adaptation is an area that has been studied and advanced over the last five years to address this problem. Two broad areas that have been studied are maximum-likelihood (ML) transformation-based adaptation (Sankar and Lee, 1994, 1996; Digalakis et al., 1995; Legetter and Woodland, 1995), and maximum a-posteriori (MAP) adaptation (Gauvain and Lee, 1994; Digalakis, Rtischev, and Neumeyer, 1996b).

Adaptation can be performed either in supervised or unsupervised mode. In supervised adaptation, transcriptions of the adaptation data are available to aid the adaptation algorithm. However, in unsupervised adaptation, transcriptions are not available. The most common approach in this case is to use a speech recognition system to produce transcriptions, which are then used by the adaptation algorithm. There is also the concept of transcription-mode adaptation. In this case, unsupervised adaptation is used on the test data, and the adapted models are used to re-recognize the test data. Transcription-mode adaptation is not causal, and the final recognition output will be delayed by at least the time-length of the data used for adaptation. Since real-time broadcast news recognition has not been a focus of Hub-4 research until recently, transcription mode adaptation is commonly used for this task.

We have used ML transformation-based adaptation in our systems. In this approach, the SI model is adapted to the test environment by means of transforms that are tied to groups of Gaussians in the HMM (Sankar and Lee, 1994, 1996; Digalakis et al., 1995; Legetter and Woodland, 1995). The transformation parameters are estimated by ML training using the adaptation data. The most common type of transforms used are a block-diagonal affine matrix transform on the HMM means and a scaling transform on the variances (Neumeyer et al., 1995). The block-diagonal mean transform usually gives most of the improvement.

Table 6 shows the improvement obtained from acoustic adaptation over the condition-specific SI models we used in the 1996 Hub-4 evaluations. This experiment uses the 1996 Hub-4 development test set. Recognition was performed using 20,000-word bigram lattices. It can be seen that mean adaptation

Condition	Models		
	Condition-specific SI	Test-cluster-adapted	
		Mean	Mean and variance
F0	22.6	21.3	20.8
F1	41.2	38.8	38.8
F2	47.2	44.0	42.3
F3	45.6	42.5	42.4
F4	36.9	34.4	33.8
F5	36.3	28.3	28.1
FX	63.8	57.8	57.2
All	41.2	37.8	37.3

Table 6: Performance of test-condition-adapted models

gives a significant improvement. Variance adaptation gives a smaller gain, though it is consistent over all the acoustic focus conditions.

To increase the speed of acoustic adaptation, we developed a new method called “basis transform adaptation” (Sankar, Gadde, and Weng, 1999). In this method, the mean transform of a speaker is represented as a weighted sum of a set of precomputed mean transforms (basis transforms). For any test speaker, only the weights need to be computed. Hence basis transform adaptation requires estimation of far fewer parameters compared to ML transformation-based adaptation using an affine matrix transform of the HMM means. The latter is also known as maximum-likelihood linear regression (MLLR). In our experiments, basis transform adaptation performed similar to MLLR for small amounts of adaptation data (Sankar et al., 1999).

We also made some engineering changes to the adaptation algorithms which resulted in significant speed improvements (Sankar et al., 1999). We used two methods, (1) Gaussian thresholding, in which we used only the Gaussians with counts higher than a threshold to estimate the transform statistics, and (2) small model adaptation, in which we used smaller models (obtained by clustering down the larger models) to estimate the adaptation transform, which was then applied on the larger models. Gaussian thresholding reduced the adaptation time by nearly 40% with no loss in accuracy. Using small model adaptation,

we obtained a 45% reduction in adaptation time with no change in accuracy for some model combinations. However, small model adaptation improvements were not consistent across all model combinations and need further study.

3.5 Toward Faster, Better and Smaller Systems

Most research for broadcast news has focussed on improving the recognition accuracy of systems. Systems with a very large number of parameters are used to increase modeling power and improve accuracy. The increased number of parameters results in increased computation during recognition, and thus slower recognition. One of our recent research goals has been to significantly increase recognition speed and decrease model size while not degrading, or even improving accuracy. Since these goals conflict with the natural tradeoff between model size or recognition speed, and accuracy, we attempted to achieve them through algorithmic approaches, significantly altering our acoustic models.

3.5.1 New Approach to Parameter Tying

Most current state-of-the-art systems use state-clustered HMM systems (Hwang, Huang, and Alleva, 1993; Woodland, Odell, Valtchev, and Young, 1994; Digalakis and Murveit, 1994). Typical systems use thousands of state clusters and over 100,000 Gaussian distributions (for example, see the Proceedings of the DARPA Broadcast News Transcription and Understanding Workshop (1998)). Since HMM state clusters have significant overlap in the acoustic space they model, Gaussians from two or more different state clusters may be used to model *the same acoustic space*. This results in two problems: (1) reduced robustness in Gaussian estimates because of data in the overlap regions being divided among the different clusters, and (2) waste of parameters because of Gaussian redundancy in the overlap regions (Sankar, 1998b).

These modeling problems can easily be addressed by decreasing the number of state clusters and correspondingly increasing the number of Gaussians per cluster. With fewer clusters, there will be less overlap between them, reducing Gaussian redundancy and improving robustness of the parameter estimates (Sankar, 1998b).

The reduced Gaussian redundancy also results in significantly fewer Gaussians being computed during recognition. We use the concept of Gaussian shortlists (Digalakis et al., 1996a) for computation efficiency. In this approach, the acoustic space is divided into Voronoi regions using vector quantization (Linde et al., 1980; Gersho and Gray, 1991). Each Voronoi region has associated with it a “shortlist” of Gaussians that have a high enough likelihood for data in this region. During recognition, the test vector is assigned to the closest Voronoi region, and only the shortlist of Gaussians in this region is used for recognition. When we decrease the number of state clusters, there is less overlap between Gaussians and thus smaller Gaussian variance, limiting their coverage of acoustic space. Thus the number of Gaussians in the shortlists decreases, resulting in a significant reduction in computation during recognition (Sankar, 1998b).

3.5.2 Per-phone Gaussian Clustering

In the PTM approach, we must use a much larger number of Gaussians per state cluster than in previous state-clustered systems. However, some phone classes may have very little acoustic variability and thus may need only a few Gaussians for good modeling. For example, the nasal */ng/* is less variable than the unvoiced stop */t/*. Thus we can use fewer Gaussians to model */ng/* and more to model */t/*.

To measure a phone’s acoustic variability, we agglomeratively cluster the HMM states for each phone, using a weighted-by-counts entropy distance between the mixture weight distributions of each state (Digalakis et al., 1996a). Clustering is stopped when the average distance reaches a prespecified relative threshold. The acoustic variability measure we used is the number of state clusters for each phone after clustering is stopped. The number of Gaussians per phone is linearly proportional to the acoustic variability of that phone, with a prespecified minimum and maximum number of Gaussians (Sankar et al., 1999; Sankar and Gadde, 1999). We also preset the acoustic variability at which the minimum and maximum number of Gaussians is realized. If the acoustic variability for a phone p is given by a_p , the minimum and maximum number of Gaussians is min_g and max_g , and the acoustic variability for the minimum and maximum number of Gaussians is min_a and max_a , respectively, then the number of Gaussians for the phone is given by

$$ng_p = \begin{cases} \min_g & \text{if } a_p < \min_a \\ \max_g & \text{if } a_p > \max_a \\ \min_g + \frac{\max_g - \min_g}{\max_a - \min_a} * (a_p - \min_a) & \text{otherwise} \end{cases} \quad (2)$$

The number of state clusters and Gaussians have been chosen to be fairly typical of that currently used in state-clustered models.

G128 : A Genone-based state-clustered model with 525 state clusters and 128 Gaussians per cluster. This represents a state-clustered system with increased tying.

P1788 : A PTM model with 39 phone classes and 1788 Gaussians per class. This represents even more tying than G128.

CLS13K : A per-phone Gaussian clustered version of PTM1788 with a total of about 13,000 Gaussians.

CLS5K : A per-phone Gaussian clustered version of PTM1788 with a total of about 5000 Gaussians.

The first three models have a similar total number of Gaussians, whereas the last two (per-phone Gaussian clustered) models have significantly fewer Gaussians.

Our first set of experiments was to measure the word error rate for each of the acoustic models, using a 48,000-word bigram language model (LM), and trigram LM lattices. The trigram lattices constrain the search space to the most likely paths for each sentence, using a previous search pass (Weng, Stolcke, and Sankar, 1998b). Thus, the word error rates observed with trigram lattices cannot necessarily be used to compare the different acoustic models. However, since lattice recognition is an important step in the multi-pass search strategy we use for our DARPA evaluation systems (Sankar et al., 1998, 1999), we decided to also compare word error rates using lattices. We used a large enough pruning beam width in the Viterbi search so that search errors did not affect the word error.

Table 7 shows that G128 and P1788 are similar in accuracy. However both are superior to the baseline G36 state-clustered model. This shows that lower word error rate can be achieved by increasing the amount of tying as compared to that used in standard state clustered systems like G36. In previous experiments on the WSJ database, we observed even larger improvements by using PTM models (Sankar, 1998b). CLS13K gives slightly better performance than P1788; however, the improvement is not statistically significant. This shows that using per-phone

3.5.3 Mixture Weight Thresholding

One problem with our PTM modeling approach is that the mixture weight distributions for each state can become very large. This occurs because our PTM system uses a much larger number of Gaussians for each phone than the number of Gaussians used for a state cluster in a typical state-clustered system. Hence each state belonging to a phone is represented by a significantly larger mixture weight distribution than a state in a state-clustered system. Since the total number of mixture weights in a model is the product of the number of mixture weights for a state multiplied by the number of states, this leads to a large storage requirement for the mixture weights. For example, a state-clustered model with 1000 state clusters, each with 32 Gaussians has the same number of Gaussians as a PTM model with 40 phones and 800 Gaussians per phone, but in terms of mixture weights the state-clustered model is 25 times smaller.

We examined two schemes to reduce the number of mixture weights that need to be represented in our PTM models (Sankar et al., 1999; Sankar and Gadde, 1999). In a “zeroing scheme”, we set all mixture weights below a threshold to zero and renormalize the mixture weights. In an “averaging scheme”, we set each mixture weight below the threshold to a value equal to the average of all mixture weights below the threshold. The mixture weights above the threshold are unchanged. Both these schemes were proposed by Gupta et al. (1996).

3.5.4 Experimental Results

We conducted an experimental study of parameter tying, per-phone Gaussian clustering, and mixture weight reduction. For training, we used the first 100 hours of Hub-4 training data, and for testing we use the 1996 Hub-4 female development test set. We trained five different acoustic models:

G36 : A Genonic model (Digalakis et al., 1996a) with 1936 state clusters and 36 Gaussians per cluster.

Gaussian clustering to reduce the number of Gaussians by more than a factor of 5 did not degrade the accuracy. However, CLS5K, which has an extremely small number of Gaussian parameters, gives a higher word error rate.

Model	Word Error Rate(%)		Number of Gaussians	Number of mixture weights (millions)
	bi	tri		
G36	40.5	31.9	69,696	0.7
G128	39.2	31.8	67,200	2.6
P1788	39.6	31.5	69,732	36.1
CLS13K	39.3	31.1	12,758	7.9
CLS5K	41.0	32.5	5,325	3.2

Table 7: Word error rates and number of parameters for different models

Next, we studied the trade-off between word error rate and recognition computation, and word error rate and recognition speed for G36, P1788 and CLS13K. The idea was to study the effect of increasing the tying over the baseline G36 model and then the effect of using the per-phone Gaussian clustering algorithm. We did this by running recognition experiments with different pruning beam widths. For each beam width, we plot the word error rate against the number of Gaussians computed per frame and the recognition time. Figure 2 shows the word error rate against the number of Gaussian distance components computed per frame, and Figure 3 shows the word error rate against recognition time when recognition is run with a bigram LM. Recognition time is given as a multiple of real time on a 400MHz Pentium II. Figure 2 shows that P1788 requires a factor of 2 fewer Gaussian computations than G36 for a word error rate of 40.5% (this is the lowest word error rate for G36). A further 2.5 factor decrease in Gaussian computation is achieved by using CLS13K. Also P1788 and CLS13K achieve this accuracy at a lower pruning threshold, thus resulting in significantly fewer active hypotheses in the search. This and the reduction in Gaussian computation result in a recognition speedup as shown in Figure 3. The figure shows that the recognition time for G36 is about 8 times real time, for P1788 about 4 times real time, and for CLS13K about 3 times real time at a word

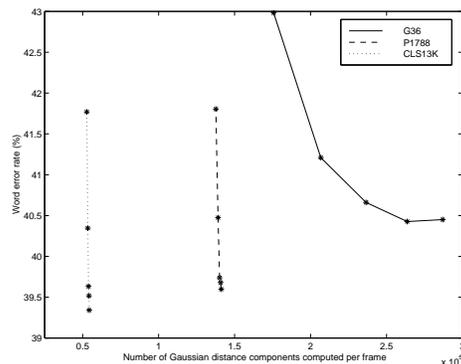


Fig. 2: Word error rate vs. number of Gaussian distance components computed

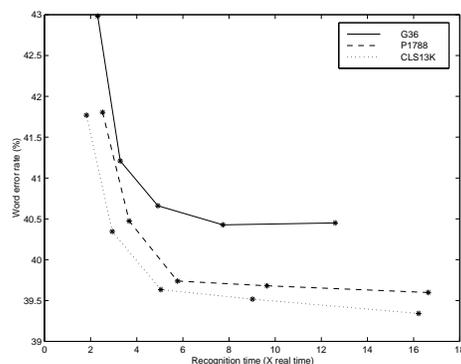


Fig. 3: Word error rate vs. Recognition time

error rate of about 40.5%. This means that a factor of 2 speedup was achieved using P1788 and a further 25% increase in speed is achieved by using CLS13K.

In the final set of experiments, we studied the effect of mixture weight reduction on the different models. This is especially important for the PTM models where the number of mixture weights is very large. Even with per-phone Gaussian clustering, the PTM models have significantly larger numbers of mixture weights than the state-clustered models, as shown in Table 7. Our experiments showed that, as the mixture weight threshold was increased, the zeroing scheme exhibited faster accuracy degradation (Sankar et al., 1999; Sankar and Gadde, 1999). In this paper, we give only the mixture weight averaging results. In the first experiment, we applied mixture weight averaging on the large PTM model (P1788) and observed the word error rate and number of mixture weights for different averaging thresholds. We plot the word error rate against the number of mixture weights in Figure 4. We see a ten-fold reduction in mixture weights is achieved with almost no degradation in accuracy. Specifically, the word error rate changes from 39.6% to 39.7% when the number of mixture weights goes from 36.2 million to 3.4 million. Further decreasing the number of mixture weights to 1.3 million increases the word error rate to 40.8%.

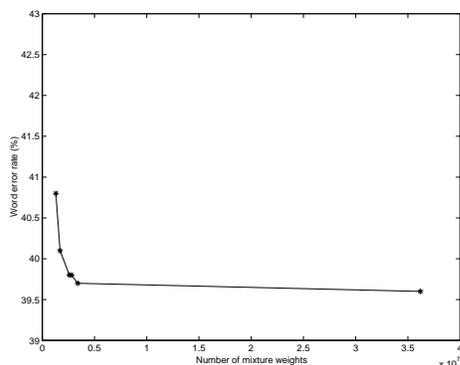


Fig. 4: Word error rate vs. number of mixture weights

The mixture weight averaging scheme can be used to reduce the mixture weights for all models (not just the PTM models). By varying the threshold individually for each model, we computed the number of mixture weights needed for each model to achieve a word error rate of about 40.5% (this is the word er-

ror rate for the baseline G36 state-clustered model). Table 8 shows these results. We see that a non-zero threshold was used for all models, showing a reduction in mixture weights for all cases. For comparison, Table 7 shows the number of mixture weights before applying the reduction algorithm. Table 8 shows that CLS13K has fewer mixture weights than the baseline, G36. It also has a five times fewer Gaussians because of per-phone Gaussian clustering. G128 has the smallest number of mixture weights. We have not applied Gaussian clustering to G128; this could result in a significant decrease in number of Gaussians as in CLS13K, and also a further decrease in the number of mixture weights. These results show that combination of Gaussian clustering and mixture weight averaging can significantly decrease model size with only a small degradation in accuracy.

Model	Threshold	Number of mixture weights (millions)	WER (%)
G36	0.01	0.6	40.3
G128	0.035	0.09	40.7
P1788	0.0025	1.5	40.5
C13K	0.01	0.40	40.5

Table 8: Word error rate (WER) for different models after mixture weight averaging

4 Language Modeling

In this section we review research aimed at improving the language models (LMs) for SRI’s Hub-4 recognition system. Over the years, we pursued essentially three fundamentally different approaches:

- **Out-of-domain data:** Exploit ancillary, non-Broadcast News data, both to increase the amount of data and to counteract deficiencies of the native broadcast news (BN) training data.
- **LM adaptation:** Adjust the LM to local stylistic and topical characteristics of the Hub-4 domain.

- **Brute force:** Add more training data and resolution (N-gram length) to the model.

The first approach proved the most successful, and was straightforward to implement using well-known techniques. The second approach, LM adaptation, is probably the most interesting one from a scientific point of view, but unfortunately met with only marginal success. The third approach is not very appealing theoretically, but did lead to appreciable improvements. More interestingly, it forced us to develop novel techniques to deal with the unwieldy increase in model size that the brute force approach entails. We will describe the three lines of research in the following sections.

4.1 Out-of-domain data

Before trying to use non-Broadcast News data in our language model, we had to deal with a mismatch between the available Hub-4 training data. The training corpus consisted of two subsets: a large (130 million word) part transcribed by commercial services, which we will designate as “H4_LM.” A second source of Hub-4 training data were the transcripts of the acoustic training corpus, referred to here as “H4_AC.” The size of this corpus was negligible relative to H4_LM (380,000 words),² but contained more detailed transcriptions following LDC standards. Spontaneous speech phenomena, such as filled pauses, fragments, and other disfluencies were much better reflected in H4_AC. This suggests that H4_AC should be weighted disproportionately to its size for best results. We therefore trained separate trigram LMs for the two Hub-4 corpora and interpolated the word probabilities from the two LMs. The mixture weight between the two models was chosen to minimize perplexity on the 1996 Hub-4 development test set. The optimal weight was determined to be 0.3 for the H4_AC model (0.7 for H4_LM), confirming the importance of the differential weighting.

We also conducted N-best rescoring experiments to assess the effect the interpolation strategy had on word error. Table 9 shows both perplexity and recognition results. Interpolating H4_AC reduces perplexity

²Unless otherwise noted, the experiments described in this section were carried out with the 1996 evaluation system at a time when only 50 hours of acoustic training data were available.

Model	PPL	WER
H4_LM trigram only	204	34.0%
H4_AC + H4_LM trigram	174	33.8%

Table 9: Effect of using acoustic training transcripts on perplexity (PPL) and word error rate (WER).

Model	PPL	WER
H4 (baseline)	174	33.8%
H4 + SWB	172	
H4 + NABN	166	
H4 + NABN + SWB	163	33.4%

Table 10: Perplexity (PPL) and word error rate (WER) using non-Hub-4 LM training material.

by 15% and lowers word error rate by 0.2%. (The effect on word error rate is probably underestimated in this experiment since the N-best lists had been generated with a bigram LM that contained both H4_LM and H4_AC training data.)

We then turned to the use of non-Hub-4 data. Specifically, we experimented with two widely available corpora: the 1995 North American Business News (NABN) LM training corpus and the Switchboard (SWB) corpus of spontaneous telephone conversations (Godfrey et al., 1992). We felt that these two corpora could complement the Hub-4 sources at opposing ends of the stylistic spectrum: NABN with written language (but subject material relevant to Hub-4) and SWB with spontaneous speech. As for the Hub-4 data, we built separate trigram models and found optimal weights for an interpolated overall LM. The resulting model had the following structure:

$$\begin{aligned}
 P(w | h) = & .64 * P(w | h, H4_LM) + \\
 & .14 * P(w | h, H4_AC) + \\
 & .16 * P(w | h, NABN) + \\
 & .06 * P(w | h, SWB)
 \end{aligned}$$

Table 10 shows the effect the four-way interpolation had on word error, using the interpolated Hub-4-only LM as a baseline. Perplexities are also shown for two (intermediate three-way interpolated LMs,

to assess the approximate impact of the individual sources. We see that the effect of adding the NABN model is appreciable (5% perplexity reduction), but that SWB adds little extra information (1-2% perplexity reduction). The latter can be attributed to three factors: the small relative corpus size of SWB (1.8M words compared to over 100M for H4 and NABN); the large stylistic difference between SWB and even the spontaneous portions of Hub-4; and the fact that Hub-4 spontaneous speech is already covered in the H4_AC material.

Overall, the straightforward interpolation approach worked surprisingly well. Compared to the model using only H4_LM data, the addition of H4_AC, NABN and SWB data lowered perplexity by 20% and word error by 0.6% absolute.

4.2 Language model adaptation

Broadcast News material is quite heterogeneous, containing a mix of speaking styles and topics. Both style and topic can be expected to exhibit local consistency, i.e., remain constant over the course of several sentences or even whole stories. This suggests that we find ways to adapt the LM to these local properties of the data.

There have been numerous attempts to explore the topical structure of Broadcast News for LM adaptation using a variety of approaches (Iyer and Ostendorf, 1996; Seymore and Rosenfeld, 1997; Clarkson and Robinson, 1997; Bellegarda, 1997, among others). Many of these approaches have realized substantial reductions in perplexity, although significant improvements in recognition accuracy have been more elusive. We obtained similar results with a topic-adaptive LM based on automatic agglomerative clustering of Broadcast News stories, reported elsewhere (Weng, Stolcke, and Sankar, 1997). While the topic-adapted model reduced perplexity over a baseline model trained from the same Hub-4 data, the improvement (including in word error) became negligible as soon as a large four-gram model based on multiple sources (as describe in the previous section) was used.

Here we focus on an different kind of LM adaptation, aimed at speaking style rather than topics. Our approach leverages the “focus conditions” by which the Hub-4 data had been classified for diagnostic purposes, as explained in Section 2. In the 1996

Model	General	By condition
H4_AC, 3gram	379	361
H4 + SWB + NABN, 4gram	154	151

Table 11: Perplexity results for condition-specific LM

Hub-4 partitioned evaluation the acoustic condition (F-condition) labels were available to the recognition system, suggesting their use for modeling. From the point of view of language modeling, the F-conditions are of interest since they are partly defined by different speaking styles, which in turn affect word choice and syntactic structure. For example, condition F0 contains planned speech while F1 consists of spontaneous speech.

Unfortunately, the acoustic model training data (H4_AC) is the only portion of the training corpus labeled for F-conditions. Even if we had ways to automatically label additional training data for F-conditions the effect would be questionable given that the other training corpora either do not contain the variety of styles found in BN (such as NABN), or do not represent them faithfully (as explained in the discussion of the H4_LM data above).

Despite these obstacles, we wanted to investigate the potential benefit of speaking style-specific LMs. We partitioned the H4_AC data by F-condition, and trained a separate trigram LM on each subset. The condition-specific LMs were then interpolated optimally with the general H4_AC trigram model.

The baseline model in this case is the general H4_AC LM by itself (which has a fairly high perplexity due to the relatively small amount of training data). The resulting model effectively gives extra weight to the data of one condition. During partitioned evaluation, we rescore each utterance using the LM matching the acoustic condition of the utterance.

The perplexity results for the Hub-4 development data are summarized in Table 11. The first row shows the baseline for this experiment, a static LM that was trained only on H4_AC data. The condition-specific LM gives a 5% perplexity reduction in this case. However, when using the same approach on a production LM (a fourgram incorporating H4_LM, NABN and SWB data), the perplexity reduction becomes marginal. This can be explained by the fact that

the condition-specific training data is several orders of magnitude smaller than the general LM training data, diluting the style-specificity of the overall model. We conclude that condition-specific data in much larger quantity would be required to improve a state-of-the-art LM.

4.3 Precise and efficient N-gram model pruning

Besides trying to leverage out-of-domain training data and to adapt the LM to subsets of the Hub-4 corpus, we also expanded the scope of our N-gram LMs over the years. Similar to other Hub-4 systems (e.g., those fielded by Cambridge University, CMU and IBM described in the Proceedings of the DARPA Broadcast News Transcription and Understanding Workshop (1998)), we moved from the traditional trigram models to four-gram and five-gram LMs, with appreciable success. For example, we found that moving from a trigram to a four-gram model reduces perplexity by about 5% relative and word error by 0.3% absolute.

As the number of potential parameters in our models grew (both as a result of high-order N-grams and because of more training data) we found it desirable to develop ways to limit the size of our models, as both disk space requirements and the time needed for loading models into memory were growing to be significant limiting factors in our experiments.

Two previous approaches address the problem of pruning parameters from N-gram models. Seymore and Rosenfeld (1996) proposed a heuristic criterion for eliminating N-grams that have little effect on the model entropy, based on their frequency in the data and the difference in N-gram log probability and the corresponding backoff estimate. Kneser (1996) explicitly aims to minimize the increase in model entropy when choosing N-grams for model pruning.

As Kneser, we chose model entropy as a principled criterion for N-gram pruning, i.e., our goal is to select N-grams such that the Kullback-Leibler (KL) divergence between the original and the pruned model is minimized. Unlike the method of Seymore and Rosenfeld (SR for short) this approach relies only on information in the model itself, without referring back to N-gram counts, which is an important convenience. However, computing the KL divergence due to even a single N-gram parameter is a non-trivial problem, since the back-off structure of the model implies that many conditional probabilities are affected

by the elimination of a single N-gram. Both previous approaches did not consider this effect to simplify the computation.

In developing our own N-gram pruning algorithm we realized that it was possible to compute the KL divergence for single N-gram pruning *exactly* and *efficiently*, i.e., with constant effort per N-gram. Below, we first describe our method, evaluate its effectiveness on Hub-4 LMs, and finally compare it to SR’s approximate method.

4.3.1 N-gram pruning by relative entropy

An N-gram language model represents a probability distribution over words w , conditioned on $(N - 1)$ -tuples of preceding words, or histories h . Only a finite set of N-grams (w, h) have conditional probabilities explicitly represented in the model. The remaining N-grams are assigned a probability by the recursive backoff rule

$$p(w|h) = \alpha(h)p(w|h')$$

where h' is the history h truncated by the first word (the one most distant from w), and $\alpha(h)$ is a *backoff weight* associated with history h , determined so that $\sum_w p(w|h) = 1$.

The goal of N-gram pruning is to remove explicit estimates $p(w|h)$ from the model, thereby reducing the number of parameters, while minimizing the performance loss. Note that after pruning, the retained explicit N-gram probabilities are unchanged, but backoff weights will have to be recomputed, thereby changing the values of implicit (backed-off) probability estimates. Thus, the pruning approach chosen is conceptually independent of the estimator chosen to determine the explicit N-gram estimates.

As mentioned above, a principled criterion to guide N-gram pruning is relative entropy or Kullback-Leibler divergence between the original and the pruned model. The relative entropy is a continuous function of the model distributions, and is zero if and only if the two distributions are identical; hence it is suitable as a pseudo-distance between models.

Let $p(\cdot|\cdot)$ denote the conditional probabilities assigned by the original model, and $p'(\cdot|\cdot)$ the probabilities in the pruned model. Then, the relative entropy

between the two models is

$$D(p||p') = - \sum_{w_i, h_j} p(w_i, h_j) [\log p'(w_i|h_j) - \log p(w_i|h_j)] \quad (3)$$

where the summation is over all words w_i and histories (contexts) h_j .

Our goal will be to select N-grams for pruning such that $D(p||p')$ is minimized. However, it would not be feasible to minimize over all possible subsets of N-grams. Instead, we will assume (as do the other pruning approaches mentioned earlier) that the N-grams affect the relative entropy roughly independently, and compute $D(p||p')$ due to each individual N-gram. We can then rank the N-grams by their effect on the model entropy, and prune those that increase relative entropy the least.

To choose pruning thresholds, it is helpful to look at a more intuitive interpretation of $D(p||p')$ in terms of *perplexity*, the average branching factor of the language model. The perplexity of the original model (evaluated on the distribution it embodies) is given by

$$PPL = e^{-\sum_{h,w} p(h,w) \log p(w|h)},$$

whereas the perplexity of the pruned model on the original distribution is

$$PPL' = e^{-\sum_{h,w} p(h,w) \log p'(w|h)}$$

The relative change in model perplexity can now be expressed in terms of relative entropy:

$$\frac{PPL' - PPL}{PPL} = e^{D(p||p')} - 1$$

This suggests a simple thresholding algorithm for N-gram pruning:

1. Select a threshold θ .
2. Compute the relative perplexity increase due to pruning each N-gram individually.
3. Remove all N-grams that raise the perplexity by less than θ , and recompute backoff weights.

4.3.2 Computing Relative Entropy

We now show how the relative entropy $D(p||p')$ due to pruning a single N-gram parameter can be computed

exactly and efficiently. Consider the effect of removing an N-gram consisting of history h and word w . This entails two changes to the probability estimates.

- The backoff weight $\alpha(h)$ associated with history h is changed, affecting all backed-off estimates involving history h . We use the notation $\text{BO}(w_i, h)$ to denote this case, i.e., that the original model does not contain an explicit N-gram estimate for (w_i, h) . Let $\alpha(h)$ be the original backoff weight, and $\alpha'(h)$ the backoff weight in the pruned model.
- The explicit estimate $p(w|h)$ is replaced by a backoff estimate

$$p'(w|h) = \alpha'(h)p(w|h')$$

where h' is the history obtained by dropping the first word in h .

All estimates not involving history h remain unchanged, as do all estimates for which $\text{BO}(w_i, h)$ is not true.

Substituting in (3), we get

$$\begin{aligned} D(p||p') &= - \sum_{w_i} p(w_i, h) [\log p'(w_i|h) - \log p(w_i|h)] \quad (4) \\ &= -p(w, h) [\log p'(w|h) - \log p(w|h)] \\ &\quad - \sum_{w_i : \text{BO}(w_i, h)} p(w_i, h) [\log p'(w_i|h) - \log p(w_i|h)] \\ &= -p(h) \left\{ p(w|h) [\log p'(w|h) - \log p(w|h)] \right. \\ &\quad \left. + \sum_{w_i : \text{BO}(w_i, h)} p(w_i|h) [\log p'(w_i|h) - \log p(w_i|h)] \right\} \end{aligned}$$

At first it seems as if computing $D(p||p')$ for a given N-gram requires a summation over the vocabulary, something that would be infeasible for large vocabularies and/or models. However, by substituting in the terms for the backed-off estimates, we see that the sum can be factored so as to allow a more efficient computation.

$$\begin{aligned} D(p||p') &= -p(h) \left\{ p(w|h) [\log p(w|h') + \log \alpha'(h) - \log p(w|h)] \right. \\ &\quad \left. + \sum_{w_i : \text{BO}(w_i, h)} p(w_i|h) [\log \alpha'(h) - \log \alpha(h)] \right\} \\ &= -p(h) \left\{ p(w|h) [\log p(w|h') + \log \alpha'(h) - \log p(w|h)] \right. \\ &\quad \left. + [\log \alpha'(h) - \log \alpha(h)] \sum_{w_i : \text{BO}(w_i, h)} p(w_i|h) \right\} \end{aligned}$$

The sum in the last line represents the total probability mass given to backoff (the numerator for computing $\alpha(h)$); it needs to be computed only once for each h , which is done efficiently by summing over all *non-backoff* estimates:

$$\sum_{w_i:\text{BO}(w_i,h)} p(w_i|h) = 1 - \sum_{w_i:\neg\text{BO}(w_i,h)} p(w_i|h)$$

The marginal history probabilities $p(h)$ are obtained by multiplying conditional probabilities $p(h_1)p(h_2|h_1) \dots$

Finally, we need to compute the revised backoff weights $\alpha'(h)$ efficiently, i.e., in constant time per N-gram. Recall that

$$\alpha(h) = \frac{1 - \sum_{w_i:\neg\text{BO}(w_i,h)} p(w_i|h)}{1 - \sum_{w_i:\neg\text{BO}(w_i,h)} p(w_i|h')}$$

$\alpha'(h)$ is obtained by dropping the term for the pruned N-gram (w, h) from the summation in both numerator and denominator. Thus, we compute the original numerator and denominator once per history h , and then add $p(w|h)$ and $p(w|h')$, respectively, to obtain $\alpha'(h)$ for each pruned w .

4.3.3 Experiments

We evaluated relative entropy-based language model pruning in SRI’s 1996 Hub-4 evaluation system. N-best lists generated with a bigram language model were rescored with various pruned versions of a large four-gram language model.

As noted in Section 4.3, the pruning algorithm is applicable irrespective of the particular N-gram estimator used. We used Good-Turing smoothing (Good, 1953) throughout and did not investigate possible interactions between smoothing methods and pruning.

Table 12 shows model size, perplexity and word error results as determined on the development test set, for various pruning thresholds. The first and last rows of the table give the performance of the full four-gram and the pure trigram model, respectively. Note that perplexity here refers to the independent test set, not to the training set perplexity that underlies the pruning criterion.

As shown, pruning is highly effective. For $\theta = 10^{-8}$, we obtain a model that is 26% the size of the

θ	bigrams	trigrams	4-grams	PPL	WER
0	11093357	14929826	3266900	163.0	32.6
10^{-9}	7751596	9634165	1938343	163.9	32.6
10^{-8}	3186359	3651747	687742	172.3	32.6
10^{-7}	829827	510646	62481	202.3	33.9
0	11093357	14929826	0	172.5	32.9

Table 12: Perplexity (PPL) and word error rate (WER) as a function of pruning threshold and language model sizes.

original model without degradation in recognition performance and less than 6% perplexity increase. Comparing the pruned four-gram model to the full trigram model, we see that it is better to include non-redundant four-grams than to use a much larger number of trigrams. The pruned ($\theta = 10^{-8}$) four-gram has the same perplexity and lower word error ($p < 0.07$) than the full trigram.

4.3.4 Comparison to Seymore and Rosenfeld’s Approach

Seymore and Rosenfeld (1996) proposed a different pruning scheme for backoff models (henceforth called the “SR criterion,” as opposed to the relative entropy, or “RE criterion”). In the SR approach, N-grams are ranked by a weighted difference of the log probability estimate before and after pruning,

$$N(w, h)[\log p(w|h) - \log p'(w|h)] \quad (5)$$

where $N(w, h)$ is the discounted frequency with which N-gram (w, h) was observed in training. Comparing (5) with the expansion of $D(p||p')$ in (4), we see that the two criteria are related. First, we can assume that $N(w, h)$ is roughly proportional to $p(w, h)$, so for ranking purposes the two are equivalent. The difference of the log probabilities in (5) corresponds to the same quantity in (4). Thus, the major difference between the two approaches is that the SR criterion does not include the effect on N-grams other than the one being considered, namely, those due to changes in the backoff weight $\alpha(h)$.

To evaluate the effect of ignoring backed-off estimates in the pruning criterion we compared the performance of the SR and the RE criterion on the Broadcast

No. Trigrams	SR	RE
1000	238.1	237.9
10000	225.1	223.9
100000	207.3	205.2
1000000	186.4	184.7

Table 13: Comparison of Seymore and Rosenfeld (SR) and Relative Entropy (RE) pruning: perplexities as a function of the number of trigrams.

No. Trigrams	SR	RE
0	35.8	
1000	35.5	35.5
10000	34.8	34.8
100000	34.3	34.2
1000000	33.2	33.1
All	32.9	

Table 14: Comparison of Seymore and Rosenfeld (SR) and Relative Entropy (RE) pruning: word error rate as a function of the number of trigrams.

News development test set, using the same N-best rescoring system as described before. To make the methods comparable we adopted Seymore and Rosenfeld’s approach of ranking the N-grams according to the criterion in question, and to retain a specified number of N-grams from the top of the ranked list. For the sake of simplicity we used a trigram-only version of the Hub-4 language model used earlier, and restricted pruning to trigrams.

We also verified that the discounted frequency $N(w, h)$ in (5) could be replaced with the model’s N-gram probability $p(w, h)$ without changing the ranking significantly: over 99% of the chosen N-grams were the same. This means the SR criterion can also be based entirely on information in the model itself, making it more convenient for model post-processing.

Tables 13 and 14 show model perplexity and word error rates, respectively, for the two pruning methods as a function of the number of trigrams in the model. In terms of perplexity, we see a very small, albeit consistent, advantage for the relative entropy method, as expected given the optimized criterion. However, the difference is negligible when it comes to recogni-

No. Trigrams	No. shared trigrams
1000	883
10000	8721
100000	85599
1000000	852016

Table 15: Overlap of selected trigrams between SR and RE methods.

tion performance, where results are identical or differ only non-significantly. We can thus conclude that, for practical purposes, the SR criterion is a very good approximation to the RE criterion.

Finally, we looked at the overlap of the N-grams chosen by the two criteria, shown in Table 15. The percentage of common trigrams ranges from 88.3% to 85.2%, and seems to decrease as the model size increases. We can expect the most frequent N-grams to be among those that are shared, thus explaining why both methods perform so similarly.

5 Improved Lattice-based Decoding

Progressive search techniques have previously been proposed as a method of gradually applying complex knowledge sources in decoding for automatic speech recognition (ASR) (Murveit et al., 1993). The strategy is exemplified in our 1996 evaluation system (Sankar et al., 1997). Non-crossword acoustic models and bigram LMs were first used to create word bigram lattices. Adapted acoustic models were then applied to produce N-best lists from these lattices. Finally, more complex knowledge sources such as crossword acoustic models and trigram LMs were added to rescore the N-best lists.

To improve our Hub-4 system performance in terms of accuracy, speed and size, we have recently introduced several enhancements in our lattice-based recognition system (Weng et al., 1998b; Weng, Stolcke, and Sankar, 1998a). These new developments include the implementation of a word-pair bigram lattice generation algorithm, two bigram lattice optimization techniques, and an efficient expansion algorithm of bigram lattices to trigram lattices.

5.1 Bigram Lattice Generation Algorithm

Our previous bigram word lattice algorithm created a subset of the full LM for an input utterance, so that the search would be narrowed in later-stage processing (Murveit et al., 1993). Unfortunately, that implementation was too simplistic in that the LM backoff node was retained in the lattice, causing almost any word in the lattice to be hypothesized at each word end, thus resulting in slow recognition. It also prevented easy expansion to trigram lattices, because the connectivity of the backoff node to almost all the other nodes in a lattice results in $O(n^2)$ trigram contexts, where n is the number of words in the lattice.

Our new bigram lattice generation algorithm is based on the word-dependent N-best algorithm (Schwartz and Austin, 1991), and is similar to those of Ney and Aubert (1994) and Odell (1995). The algorithm assumes that the starting time for a word depends on the preceding word but not on any word before that. Thus, a separate word hypothesis is propagated for each possible predecessor word. When the word ends, the score of each word ending hypothesis along with the previous word is recorded in a backtrace array. The best hypothesis is propagated along with the current word label.

To generate a lattice, we process the backtrace array to cluster together words that have the same name and ending time in the search. Each such cluster corresponds to a node in the lattice. If the word corresponding to cluster A starts the word corresponding to cluster B in the search, then node A is connected to node B in the lattice. For a better control of lattice sizes as well as their quality, we use forward-backward search (Austin, Schwartz, and Placeway, 1991) and produce bigram lattices in the backward pass.

Table 16 gives recognition results on F0, F1, and FX conditions for the male speakers of the 1996 Hub-4 development set, using previous and new lattice generation algorithms to generate bigram lattices. The lattices were used to create N-best lists, which were then rescored using SRI’s 1996 Hub-4 48K vocabulary trigram LM (Weng et al., 1997). The acoustic models used for recognition are SRI’s 1996 adapted models (Sankar et al., 1997). The table shows that the new algorithm gives only a small 1.8% relative improvement over the old algorithm when rescored with a trigram LM. However, we observed an order of magnitude speedup by using the new bigram lattices

Lattices	Resc. LM	F0	F1	FX	Sub-total
Old bigram	trigram	14.1	35.1	60.1	33.5
New bigram	trigram	13.9	34.5	59.1	32.9

Table 16: Comparison of results of the previous and new bigram lattice algorithms.

for recognition as compared to the previous bigram lattices.

5.2 Lattice Reduction

For acoustically degraded speech, the bigram lattices generated in the forward-backward recognition pass are quite large. For subsequent decoding passes, we expand the bigram lattices to trigram lattices (see Section 5.3). Trigram expansion significantly increases the lattice size. Thus if the bigram lattices are too large, it will not be feasible to expand them.

One way to obtain smaller bigram lattices is to lower the backward pruning threshold. However, this results in an undesirable increase in the lattice word error rate (the error rate for the best path in the lattice). To reduce lattice size without increasing lattice error, we developed a lattice reduction algorithm. In it, we combine identical or near identical (sub)paths in the lattices so that the redundant nodes and transitions are removed. Related methods are well known from the computer science literature (Hopcroft and Ullman, 1979), where standard algorithms for minimizing deterministic finite state automata (FSA) are given. More recently, algorithms for minimization of weighted transducers have been developed (Mohri and Riley, 1997; Strom, 1997). The key difference between these approaches and ours is that our lattices are nondeterministic and there is no requirement that the resulting lattices be deterministic.

The main observation underlying our first algorithm (Weng et al., 1998b) is that if two nodes in the lattice have the same word label and the same set of successor (or predecessor) nodes, they can be merged without changing the language of the lattice, where the language of a lattice is defined as the set of all the word strings starting at the initial node and ending at the final node. Depending on whether we are merging nodes according to their predecessor node set or their

successor node set, we can have either a forward or a backward version of the reduction algorithm. Multiple iterations can also be performed. Here we only describe the backward reduction algorithm; the forward one is symmetrical in structure.

5.2.0.1 Backward

Reduction

Algorithm:. Let $S_{out}(n)$ and $S_{in}(n)$ be the set of successor nodes and the set of predecessor nodes of node n , respectively, and $word(n)$ be the word name of node n .

- For each lattice node n in reverse topological order (starting with the final node):
 - for each pair of predecessor nodes (i, j) of node n :
 - * if $word(i) = word(j)$ and $S_{out}(i) = S_{out}(j)$, then merge nodes i and j

Our second lattice reduction algorithm (Weng et al., 1998a) relaxes the restriction of requiring $S_{out}(i) = S_{out}(j)$ in the algorithm and resulted in improved lattice error rates as well as smaller lattice sizes. Instead of $S_{out}(i) = S_{out}(j)$, only a certain percentage of overlap between the two outgoing/incoming node sets is required for node merging. This algorithm produces smaller lattices and adds new hypotheses to the lattices. Therefore, it is an approximate reduction algorithm, as opposed to the previous exact reduction algorithm.

We evaluated the effectiveness of the reduction algorithm on lattices generated from the 1996 DARPA Hub-4 development test set, using an unadapted version of SRI’s 1997 acoustic models (Sankar et al., 1998). Only the F0 and F1 conditions of that set were included, with F1 generally giving considerably larger lattices.

For comparison, we also performed reduction experiments using a standard FSA determinization/minimization approach. We first converted our node-based word lattices into the dual FSA representation, a process which maps each node to exactly one FSA transition. We then performed FSA determinization and minimization using the AT&T FSM Toolkit (Mohri, Pereira, and Riley, 1998). Since bigram probabilities can always be retrofitted into a word lattice without changing its structure, we first

set all transition probabilities to 1, effectively turning the weighted FSA operations into their classical, non-weighted counterparts. For comparison purposes, we then transformed the minimized FSA back into a node-based word lattice.³

From the results shown in Table 17, we observed that the exact and approximate reduction algorithms gave a 50% and 67% size reduction, respectively, over the original bigram lattices. Size is measured as the average number of transitions per lattice. The approximate reduction algorithm also gave a 6% and 34% lower lattice error for F0 and F1 conditions. Compared with the standard finite state machine (FSM) determinization and minimization algorithms implemented by AT&T, our two algorithms produced lattices with 8% and 39% smaller sizes.

	Average Number of transitions		Lattice Error Rate	
	F0	F1	F0	F1
Baseline	11624	16208	3.3%	10.0%
FSM Det/Min	6417	8715	3.3%	10.0%
Exact Red	6129	7797	3.3%	10.0%
Approx Red	4318	4985	3.1%	6.6%

Table 17: Sizes and lattice error rates of the reduction algorithms

The above experiments also show that both non-deterministically reduced and determinized/minimized lattices gave virtually identical recognition times. Furthermore, in spite of the lower lattice error rate from the approximate reduction algorithm, we observed no improvement on 1-best recognition.

5.3 Algorithms for Expansion to Trigram Lattices

In previous work, we had incorporated trigram LMs by rescoreing N-best lists (Sankar et al., 1997). By incorporating trigrams earlier in the multi-pass decoding process, we would hope to obtain better accuracy. We incorporated trigram LMs into lattice decoding

³The reverse conversion constructs a node for each unique pair of FSA node and incoming transition symbol. This produces best results if the FSA is deterministic. Conversions back and forth between the two representations are designed to be exact inverses.

by developing and implementing algorithms to expand bigram lattices to trigram lattices (Weng et al., 1998b, 1998a). For simplicity and concreteness, our discussion here is focussed on trigram lattices, but the algorithms described generalize to N-gram models of higher order.

Using a conventional approach to place trigram probabilities on the lattice transitions, a unique two-word context for each transition must be created. We observed that this leads to a ten-fold increase of the number of lattice transitions on our Hub-4 development set. We therefore developed a compact expansion algorithm (Weng et al., 1998b) that takes advantage of the backoff structure of the N-gram model. For most trigram language models, the number of explicit trigrams is much smaller than the number of all possible trigrams. Furthermore, as shown in Section 4.3, most trigrams can be eliminated without loss of performance. If we can share the bigram backoff weights for trigram contexts, then we need to duplicate only enough nodes to uniquely represent the explicit trigram probabilities in the lattice.

Figure 5 illustrates the differences between alternate trigram expansion algorithms, given that there is only one explicit trigram probability $p(d|ac)$, where c is the nearest context word. Figure 5a shows the unexpanded bigram lattice. Figure 5b depicts the result of the conventional trigram expansion method. Notice that there are three copies of node c , one for each unique predecessor word. Finally, Figure 5c shows the result of our new compact trigram expansion algorithm, which creates one copy of node c only for each explicit trigram involving c and the words adjacent to it in the lattice. Therefore, only one node labeled c and its incoming transition from the node labeled a and outgoing transition to the node labeled d are created. The explicit trigram probability $p(d|ac)$ is placed on the transition from the newly created node to the node labeled d . The weight on the transition from the node labeled a to the newly created node was copied from the weight on the transition from the node labeled a to the original node labeled c . After the explicit trigrams are processed, the outgoing transitions from the original node labeled c are weighted with their corresponding bigram probabilities $p(d|c)$ and $p(e|c)$. Furthermore, bigram backoff weights $bo(a, c)$, $bo(b, c)$, and $bo(f, c)$ are multiplied onto the corresponding incoming transitions of the

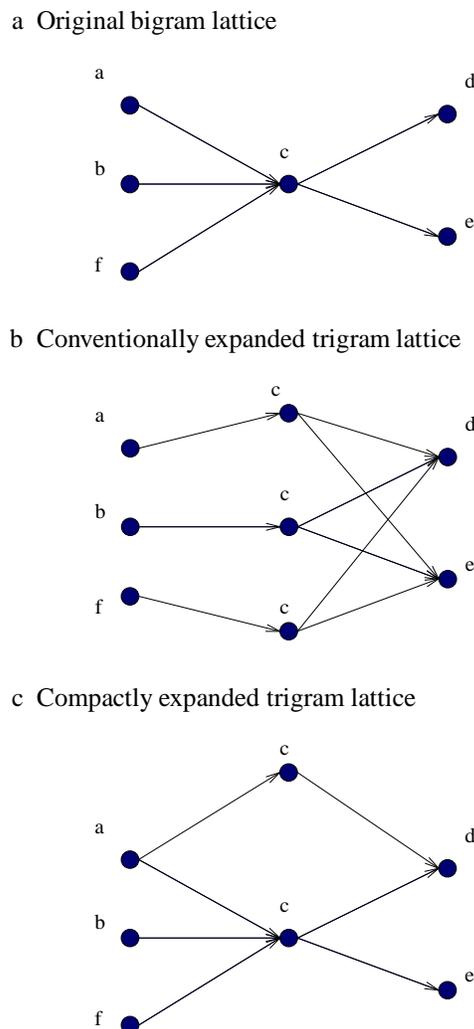


Fig. 5: Expansion of bigram to trigram lattice.

original node labeled c . Therefore, taking the path a, c, d would result in a backed-off trigram probability $p(d|ac) = p(d|c) * bo(a, c)$, in addition to the weight previously placed on transition (a, c) .

5.3.0.2 Compact lattice expansion algorithm: Let $weight(i, j)$ be the aggregate probability on transition (i, j) .

For each node n in the lattice in topological order:

- for each predecessor node i of n :
 - for each successor node k of n :
 - * if there is an explicit trigram probability for $(word(i), word(n), word(k))$,
 - if a node j with $word(n)$ was already created for trigram context $(word(i), word(n))$ and $word(k)$, connect node i to node j
 - otherwise, create node j , label it with $word(n)$, connect node i to node j and node j to node k , and set $weight(j, k) = p(word(k)|word(i)word(n))$
 - * otherwise, mark transitions (i, n) and (n, k)
 - if transition (i, n) is not marked, remove (i, n) ; otherwise, set $weight(i, n) = weight(i, n) * bo(word(i), word(n))$
- for each end successor node k of n :
 - if transition (n, k) is not marked, remove (n, k) ;
 - otherwise, set $weight(n, k) = p(word(k)|word(n))$
- if no incoming transitions are marked, remove node n and all its incoming and outgoing transitions.

A potential problem for the compact algorithm is that even for explicit trigram probabilities, the lattice retains a path using the backoff transitions, which might have a higher weight than the correct trigram transition and therefore be preferred during search.

	F0	F1	FX	Sub-Total
Nbest resc w. new biLat.	13.9	34.5	59.1	32.9
Trigram exp. w. new biLat.	13.5	33.1	57.9	31.9

Table 18: Improvement with new lattice algorithms.

For example, in Figure 5(b), there are two paths labeled (a, c, d) , and during search the incorrect lower path will be chosen if $p(d|ac) < p(d|c) * bo(ac)$. Our solution to this problem is to preprocess the trigram LM to eliminate all trigram probabilities that are lower than the corresponding (improper) backoff estimate, and to renormalize the LM. However, in practice, this only eliminates a small fraction of trigrams without affecting recognition accuracy.

We conducted two sets of experiments. In the first, we compared recognition directly from conventionally expanded trigram lattices to rescoring N-best lists with trigram LMs. Table 18 gives recognition word error rates for the male F0, F1, and FX conditions of the 1996 Hub-4 development data with our 1996 adapted acoustic models (Sankar et al., 1997). The first row corresponds to generating bigram lattices with our new lattice generation algorithm, creating N-best lists from these lattices, and then rescoring the N-best lists with trigram LMs. The second row corresponds to recognition directly from trigram lattices, conventionally expanded from the bigram lattices. The table shows that a 3% relative improvement is obtained by decoding with a trigram lattice as opposed to using trigram LMs to rescore N-best lists.

The second set of experiments is to examine the effectiveness of our new compact expansion algorithm, using the reduced bigram lattices described in Section 5.2. We expanded the reduced bigram lattices using both the conventional and compact trigram lattice expansion algorithms. We found that the compact expansion algorithm was ten times faster than the conventional algorithm. Furthermore, Table 19 shows that the size of the trigram lattices from the compact expansion algorithm is only about a sixth of those from the conventional expansion algorithm. In experiments using SRI’s unadapted acoustic models (Sankar et al., 1998), recognition accuracy with compact lattices was identical to that with conventional lattices, as expected.

Expansion Alg.	F0	F1	Sub-total
Conventional	123107	488738	319985
Compact	29113	76396	54573

Table 19: Trigram lattice sizes in terms of average number of transitions.

6 Summary

In our work on broadcast news recognition, we made significant improvements in three main areas: acoustic modeling, language modeling, and lattice-based decoding.

We developed and studied various acoustic modeling techniques, including methods to robustly estimate HMMs, new parameter tying techniques, Gaussian clustering and mixture weight thresholding schemes, and acoustic adaptation. The GMS algorithm and the T²-HMM algorithm gave robust parameter estimates and improved word error rate. Parameter tying studies showed that increased tying gives lower word error rate, and a significant reduction in computation during recognition. Per-phone Gaussian clustering dramatically reduced the the number of Gaussians in the system with no degradation in accuracy. Similarly, mixture weight thresholding was found to drastically reduce the number of non-zero mixture weights. Finally, we demonstrated the effectiveness of acoustic adaptation to automatically determined clusters of test segments.

Our language modeling research on broadcast news followed three main directions. The most significant improvements came from interpolating separate LMs representing different transcription styles (for acoustic vs. LM training) as well as out-of-domain data. Further improvements came from expanding the scope of our N-grams. This approach as well as the increase in training data lead us to develop a new N-gram parameter selection algorithm based on relative entropy distance, which drastically reduced model size without performance degradation. We also explored LM adaptation to topic and F-condition changes; these approaches showed promise in the form of perplexity reductions, but so far no improvements in recognition accuracy.

In our work on lattice-based decoding, we devel-

oped a new bigram lattice generation algorithm which gave lower recognition word error rate than our previous algorithm at drastically increased recognition speed. We also developed a new bigram lattice reduction algorithm that gives more than a factor of two reduction in lattice size. Recognition directly from trigram lattices was shown to be significantly better than rescoring N-best lists with trigram LMs. Finally, we developed a new compact trigram expansion algorithm than gives a six-fold reduction in lattice size compared to a conventional trigram expansion algorithm.

References

- Austin, S., Schwartz, R., and Placeway, P. (1991). The forward-backward search algorithm. In *Proceedings IEEE International Conference on Acoustics, Speech, and Signal Processing* (pp. 697–700).
- Bellegarda, J. R. (1997). A latent semantic analysis framework for large-span language modeling. In G. Kokkinakis, N. Fakotakis, and E. Dermatas (Eds.), *Proceedings of the 5th European Conference on Speech Communication and Technology* (Vol. 3, pp. 1451–1454). Rhodes, Greece.
- Clarkson, P., and Robinson, A. J. (1997). Language model adaptation using mixtures and an exponentially decaying cache. In *Proceedings of the IEEE Conference on Acoustics, Speech, and Signal Processing*. Munich.
- Dempster, A., Laird, N., and Rubin, D. (1977). Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society*, 39(1), 1–38.
- Digalakis, V., and Murveit, H. (1994). High-accuracy large-vocabulary speech recognition using mixture tying and consistency modeling. In *Proceedings of the DARPA Human Language Technology Workshop*. Plainsboro, NJ.
- Digalakis, V., Rtischev, D., and Neumeyer, L. (1995). Speaker adaptation using constrained reestimation of Gaussian mixtures. *IEEE Transactions on Speech and Audio Processing*, 3(5), 357–366.
- Digalakis, V., Monaco, P., and Murveit, H. (1996a). Genones: Generalized mixture tying in continuous hidden Markov model-based speech recognizers. *IEEE Transactions on Speech and Audio Processing*, 4(4), 281–289.
- Digalakis, V., Rtischev, D., and Neumeyer, L. (1996b). Speaker adaptation using combined transformation and bayesian methods. *IEEE Transactions on Speech and Audio Processing*, 4(4), 294–300.
- Doddington, G. (1992). CSR corpus development. In *Proceedings of the DARPA SLS Workshop* (pp. 363–366).
- Gauvain, J., and Lee, C.-H. (1994). Maximum *a posteriori* estimation for multivariate Gaussian mixture observations of Markov chains. *IEEE Transactions on Speech and Audio Processing*, 2(2), 291–298.

- Gersho, A., and Gray, R. M. (1991). *Vector Quantization and Signal Compression*. Kluwer Academic Publishers.
- Godfrey, J. J., Holliman, E. C., and McDaniel, J. (1992). SWITCHBOARD: Telephone speech corpus for research and development. In *Proceedings of the IEEE Conference on Acoustics, Speech, and Signal Processing* (Vol. 1, pp. 517–520). San Francisco.
- Good, I. J. (1953). The population frequencies of species and the estimation of population parameters. *Biometrika*, 40, 237–264.
- Gupta, S., Soong, F., and Hami-Cohen, R. (1996). Quantizing mixture weights in a tied-mixture HMM. In *Proceedings of ICSLP* (pp. 1828–1831).
- Heck, L., and Sankar, A. (1997). Acoustic clustering and adaptation for robust speech recognition. In *Proceedings of EUROSPEECH*.
- Hopcroft, J., and Ullman, J. (1979). *Introduction to Automata Theory, Languages, and Computation*. Reading, MA: Addison-Wesley Publishing Company, Inc.
- Hwang, M.-Y., Huang, X., and Alleva, F. (1993). Predicting unseen triphones with Senones. In *Proceedings IEEE International Conference on Acoustics, Speech, and Signal Processing* (pp. II–311 – II–314).
- Iyer, R., and Ostendorf, M. (1996). Modeling long distance dependencies in language: Topic mixtures vs. dynamic cache models. In H. T. Bunnell and W. Idsardi (Eds.), *Proceedings of the International Conference on Spoken Language Processing* (Vol. 1, pp. 236–239). Philadelphia.
- Juang, B.-H. (1985). Maximum-likelihood estimation for mixture multivariate stochastic observations of Markov chains. *AT&T Technical Journal*, 64(6), 1235–1249.
- Kneser, R. (1996). Statistical language modeling using a variable context length. In H. T. Bunnell and W. Idsardi (Eds.), *Proceedings of the International Conference on Spoken Language Processing* (Vol. 1, pp. 494–497). Philadelphia.
- Kubala, F., Jin, H., Matsoukas, S., Nguyen, L., Schwartz, R., and Makhoul, J. (1997). The 1996 BBN Byblos Hub-4 transcription system. In *Proceedings of the DARPA Speech Recognition Workshop*. Chantilly, VA.
- Leggetter, C. J., and Woodland, P. C. (1995). Flexible speaker adaptation using maximum likelihood linear regression. In *Proceedings of the Spoken Language Systems Technology Workshop* (pp. 110–115).
- Linde, Y., Buzo, A., and Gray, R. (1980). An algorithm for vector quantizer design. *IEEE Transactions on Communications*, COM-28, 84–95.
- Mohri, M., and Riley, M. (1997). Weighted determinization and minimization for large vocabulary speech recognition. In *Proceedings of Eurospeech-97*.
- Mohri, M., Pereira, F., and Riley, M. (1998). *FSM Library—General-purpose finite-state machine software tools, Version 3.6*. <http://www.research.att.com/sw/tools/fsm/>.
- Murveit, H., Butzberger, J., Digalakis, V., and Weintraub, M. (1993). Large-vocabulary dictation using SRI’s DECIPHER (TM) speech recognition system: Progressive-search techniques. In *Proceedings IEEE International Conference on Acoustics, Speech, and Signal Processing* (pp. II 319–322).
- Neumeyer, L., Sankar, A., and Digalakis, V. (1995). A comparative study of speaker adaptation techniques. In *Proceedings of EUROSPEECH* (pp. 1127–1130).
- Ney, H., and Aubert, X. (1994). A word graph algorithm for large vocabulary, continuous speech recognition. In *Proceedings of International Conference of Spoken Language Processing*.
- Odell, J. (1995). *The Use of Context in Large Vocabulary Speech Recognition*. Unpublished doctoral dissertation, University of Cambridge.
- Proceedings of the DARPA Broadcast News Transcription and Understanding Workshop*. (1998). Lansdowne, VA.
- Sankar, A. (1998a). Experiments with a Gaussian merging-splitting algorithm for HMM training for speech recognition. In *Proceedings of the Broadcast News Transcription and Understanding Workshop*. Lansdowne, VA.
- Sankar, A. (1998b). A new look at HMM parameter tying for large vocabulary speech recognition. In *Proceedings of ICSLP*. Sydney, Australia.
- Sankar, A. (1998c). Robust HMM estimation with Gaussian merging-splitting and tied-transform HMMs. In *Proceedings of ICSLP*. Sydney, Australia.
- Sankar, A., and Gadde, V. R. R. (1999). Parameter tying and Gaussian clustering for faster, better, and smaller speech recognition. In *Proceedings of EUROSPEECH*.
- Sankar, A., and Lee, C.-H. (1994). Stochastic matching for robust speech recognition. *IEEE Signal Processing Letters*, 1(8), 124–125.
- Sankar, A., and Lee, C.-H. (1996). A maximum-likelihood approach to stochastic matching for robust speech recognition. *IEEE Transactions on Speech and Audio Processing*, 4(3), 190–202.
- Sankar, A., Heck, L., and Stolcke, A. (1997). Acoustic modeling for the SRI Hub4 partitioned evaluation continuous speech recognition system. In *Proceedings of the DARPA Speech Recognition Workshop*. Chantilly, VA.
- Sankar, A., Weng, F., Rivlin, Z., Stolcke, A., and Gadde, R. (1998). Development of SRI’s 1997 broadcast news transcription system. In *Proceedings of the Broadcast News Transcription and Understanding Workshop*. Lansdowne, VA.
- Sankar, A., Gadde, R. R., and Weng, F. (1999). SRI’s 1998 broadcast news system – toward faster, smaller, better speech recognition. In *Proceedings of the DARPA Broadcast News Workshop*. Washington, D.C.
- Schwartz, R., and Austin, S. (1991). A comparison of several approximate algorithms for finding multiple (N-best) sentence hypotheses. In *Proceedings IEEE International Conference on Acoustics, Speech, and Signal Processing* (pp. 701–704).
- Seymore, K., and Rosenfeld, R. (1996). Scalable backoff language models. In H. T. Bunnell and W. Idsardi (Eds.), *Proceedings of the International Conference on Spoken Language Processing* (Vol. 1, pp. 232–235). Philadelphia.

- Seymore, K., and Rosenfeld, R. (1997). Using story topics for language model adaptation. In G. Kokkinakis, N. Fakotakis, and E. Dermatas (Eds.), *Proceedings of the 5th European Conference on Speech Communication and Technology* (Vol. 4, pp. 1987–1990). Rhodes, Greece.
- Stern, R. (1997). Specification of the 1996 Hub4 broadcast news evaluation. In *Proceedings of the DARPA Speech Recognition Workshop* (pp. 7–10). Chantilly, VA.
- Strom, N. (1997). *Automatic Continuous Speech Recognition with Rapid Speaker Adaptation for Human/Machine Interaction*. Unpublished doctoral dissertation, KTH.
- Weng, F., Stolcke, A., and Sankar, A. (1997). Hub-4 language modeling using domain interpolation and data clustering. In *Proceedings of the DARPA Speech Recognition Workshop* (pp. 147–151). Chantilly, VA.
- Weng, F., Stolcke, A., and Sankar, A. (1998a). Efficient lattice representation and generation. In *Proceedings of ICSLP*. Sydney, Australia.
- Weng, F., Stolcke, A., and Sankar, A. (1998b). New developments in lattice-based search strategies in SRI's H4 system. In *Proceedings of the Broadcast News Transcription and Understanding Workshop*. Lansdowne, VA.
- Woodland, P., Odell, J., Valtchev, V., and Young, S. (1994). Large vocabulary continuous speech recognition using HTK. In *Proceedings IEEE International Conference on Acoustics, Speech, and Signal Processing* (pp. II-125 – II-128).
- Woodland, P. C., Hain, T., Johnson, S., Niesler, T., Tuerk, A., Whittaker, E., and Young, S. (1998). The 1997 HTK broadcast news transcription system. In *Proceedings of the Broadcast News Transcription and Understanding Workshop*. Lansdowne, VA.
- Young, S., and Woodland, P. (1993). The use of state tying in continuous speech recognition. In *Proceedings of EUROSPEECH* (pp. 2203–2206).