Online Speaker Adaptation and Tracking for Real-Time Speech Recognition

Daben Liu, Daniel Kiecza, Amit Srivastava, Francis Kubala

BBN Technologies
50 Moulton Street, Cambridge, MA 02138
{dliu,dkiecza,asrivast,fkubala}@bbn.com

Abstract

This paper describes a low-latency online speaker adaptation framework. The main objective is to apply fast speaker adaptation to a real-time (RT) large vocabulary continuous speech recognition (LVCSR) engine. In this framework, speaker adaptation is performed on speaker turns generated by online speaker change detection and speaker clustering. To maximize long-term system performance, the adaptation statistics for known speakers are updated continuously while new speakers are actively discovered. In contrast to existing approaches, a re-decode of an utterance after adaptation is eliminated from the process. We demonstrate that the framework can be easily incorporated into every pass of a multi-pass decoder. We applied the framework to the BBN Audio Indexer which is a real-time end-to-end audio indexing system that runs at around 0.6xRT. The result is an 8%-12% relative word-error-rate reduction on broadcast news benchmark tests for English, Chinese, and Arabic, with less than 0.1xRT cost in real-time throughput.

1. Introduction

In many application domains, such as broadcast news (BN), teleconferencing, or call center, the speaker characteristics are unknown. The potential mismatch in speakers between training and operation poses one of the major challenges to a statistical speech recognition system. To address this problem, speaker adaptation techniques have been developed and proven to be effective.

This paper focuses on applications where the input is a continuous audio stream, and where the overall system operation must be below real-time with an average latency of less than 1 minute. These assumptions impose a set of operational constraints on speaker adaptation algorithms:

1. Unsupervised operation.
2. Below real-time throughput.
3. Low latency.
4. Low memory footprint.

Existing speaker adaptation techniques fail to address these requirements in several ways:

1. Most adaptation techniques rely on offline speaker clustering which assumes availability of all the data. This assumption leads to high latency.
2. A post-adaptation re-decode of the same data is usually required. Since decoding is generally the most computationally expensive process, adding another decoding pass for adaptation is not practical.
3. The memory size required to store one adapted model is typically large. Storing an adapted model for every known speaker is not feasible in practice.

A number of online speaker/condition adaptation approaches have been reported in the literature. Digalakis [1] proposed an incremental version of EM algorithm to be used in online adaptation. Mokbel [2] defined a unified framework for both Maximum *a Posteriori* (MAP) and Maximum Likelihood Linear Regression (MLLR) adaptation. However, in both cases, speaker distribution appears to be given *a priori* rather than detected automatically. Zhang, *et al* [3], proposed a solution that operates incrementally by detecting speaker changes. In their approach, each utterance is decoded against a speaker-independent (SI) model and a number of speaker-adaptive (SA) models to decide which model to use. If the SI model is picked, a new SA model is created. Otherwise, the selected SA model is updated. Then the utterance is re-decoded with the new model. To reduce computation, only the models from the 2 most recent known speakers are preserved.

We have developed an online speaker tracking system that combines fast speaker change detection [4] and online speaker clustering [5]. As the front-end, the speaker tracking system enabled us to develop an efficient online speaker adaptation framework that is integrated into every pass of our multi-pass decoder and take advantage of data from many known speakers.

2. Speaker Tracking

We define speaker tracking as a two-step operation: speaker change detection (SCD) and speaker clustering.

2.1. Speaker Change Detection

We presented a phone-class based SCD algorithm in [4]. The algorithm creates a phone-class transcription of the audio stream. Phone classes are broad definitions of vowel, consonant, and non-speech event types. Then a generalized likelihood ratio (GLR) hypothesis test is applied on each phone-class boundary. A detailed description of a fast SCD implementation is presented in [4]. The output of this process is a sequence of audio segments homogeneous in speaker, which we call "speaker turns".

The hypothesis test is biased to favor boundaries on non-speech classes. It is also biased towards higher false acceptance, since falsely accepted boundaries can be rejected during speaker clustering.

A commonly used alternative method works by partitioning the audio stream into small uniform chunks before clustering. To make sure that each chunk is pure, the chunk size must be small, usually between 0.25 and 2 seconds. SCD has two advantages over this alternative. Firstly, SCD creates longer segments and thus provides more data for speaker clustering, leading to a more robust model estimation. Secondly, the boundaries detected with SCD are much less likely to occur within a word, a significant benefit for the downstream speech recognition process.

2.2. Online Speaker Clustering

We introduced an online speaker clustering algorithm in [5]. The novelty of the algorithm is that it can determine the cluster label of a speaker turn immediately as it becomes available, without relying on any future data.

The algorithm determines on the fly whether a speaker turn belongs to one of the known clusters or whether a new cluster

must be created. The computational complexity of online speaker clustering is much lower than commonly used hierarchical offline speaker clustering technique. Yet, the online version performs on par with the offline algorithm in terms of cluster purity, which is important for the purpose of adaptation. A more complete algorithm description and performance comparison is given in [5].

2.3. Online Speaker Adaptation

The speaker tracking system is created by combining SCD and online speaker clustering. The output is a sequence of audio segments that are homogeneous in speaker. Each segment is attributed with a speaker cluster label that represents a unique speaker. For decoding efficiency, speaker turns are chunked into on average 4 second long utterances by chopping on silence. Thus the output of the speaker tracking front-end is a sequence of cluster-labeled utterances.

The cluster label for each utterance serves as speaker identity that is necessary for speaker adaptation. Every utterance is used to adapt subsequent utterances of the same cluster. By relying on cluster labels, we can enrich speaker adaptation parameters causally with all past speaker turns of the same cluster. This leads to a more robust estimation of the adaptation parameters.

3. Online Speaker Adaptation

3.1. MLLR vs. MAP

There are two widely used techniques for speaker adaptation: MLLR and MAP. Many reported results have shown that MAP adaptation outperforms MLLR adaptation, when there is sufficient adaptation data [6]. However, operationally, MAP suffers the following drawbacks:

1. MAP can only adapt the model parameters for which data is available. Thus it is difficult to apply adaptation parameters trained from one utterance on a different utterance with MAP.
2. As reported in [6], MAP converges slower than MLLR with small amount of data.
3. MAP generally requires a larger amount of model parameters than MLLR, which leads to greater computational complexity.

As a result, we based our online speaker adaptation framework on MLLR.

3.2. Feature vs. Model-Based Transformation

An LVCSR system typically requires a large model space. One of our real-time LVCSR system decoding passes uses up to 64 Gaussian mixture models (GMM) for around 3000 quinphone models. The required memory space easily reaches 80MB. A model-based transformation requires a new adapted model for each speaker, which creates a significant demand for memory.

A feature-based transformation does not have the same demand for memory. Constrained model-space transformation is one form of MLLR which assumes that the model mean and variance share the same transformation matrix [7]. With this assumption, the adaptation transformation can be applied directly to the feature vector $o(\tau)$ at time τ :

$$\hat{o}(\tau) = \mathbf{A}o(\tau) + \mathbf{b} = \mathbf{W}\zeta(\tau) \tag{1}$$

where $\mathbf{A}$ is the transformation matrix, $\mathbf{b}$ is the constant bias, and $\mathbf{W}$ is the extended transformation matrix $[\mathbf{b}^T, \mathbf{A}^T]^T$, $\zeta(\tau)$ is the extended feature vector $[1 \;\; o(\tau)^T]^T$.

3.3. Optimized Feature Transformation

As demonstrated in [7], the optimal solution for the ith row $\mathbf{w}_i$ of $\mathbf{W}$ is given by:

$$\mathbf{w}_i = (\alpha\mathbf{p}_i + \mathbf{k}_i)\mathbf{G}_i^{-1} \tag{2}$$

where i denotes the ith dimension of extended feature vector, $\mathbf{p}_i$ is the extended cofactor row vector [0 c_{i1}… c_{in}], (c_{ij}=cof($\mathbf{A}_{ij}$)), $\mathbf{k}_i$ is the sufficient statistics for mean of the ith dimension and $\mathbf{G}_i$ is the sufficient statistics for variance, α is the free parameter that satisfies a quadratic function, the full solution of which is given in [7].

As formula (2) shows, $\mathbf{k}_i$ and $\mathbf{G}_i$ are all that is needed to estimate the transformation matrix. The formulae are as follows:

$$\mathbf{G}_i = \sum_{m=1}^{M} \frac{1}{\sigma_i^{(m)2}} \sum_{\tau=1}^{T} \gamma_m(\tau)\zeta(\tau)\zeta(\tau)^T \tag{3}$$

$$\mathbf{k}_i = \sum_{m=1}^{M} \frac{1}{\sigma_i^{(m)2}} \mu_i^{(m)} \sum_{\tau=1}^{T} \gamma_m(\tau)\zeta(\tau)^T \tag{4}$$

where $\sigma_i^{(m)}$ is the variance of the mth Gaussian component at the ith feature dimension, $\mu_i^{(m)}$ is the mean of the mth Gaussian component at the ith feature dimension, and $\gamma_m(\tau)$ is the posterior of the mth Gaussian component:

$$\gamma_m(\tau) = \frac{w_m N(o(\tau); \boldsymbol{\mu}_m, \boldsymbol{\Sigma}_m)}{\sum_{k=1}^{M} w_k N(o(\tau); \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)} \tag{5}$$

where N(.) denotes a Gaussian distribution, and w_k are the mixture weights.

As we can see, both $\mathbf{k}_i$ and $\mathbf{G}_i$ can be enriched incrementally as additional data becomes available. For every utterance, the adaptation can start from a set of $\mathbf{k}_i$ and $\mathbf{G}_i$ that are already accumulated over all previous utterances of the same cluster. This provides an efficient way of preserving the speaker specific information for all known speaker clusters.

The storage sizes of $\mathbf{k}_i$ and $\mathbf{G}_i$ depend on the structure of the transformation matrix. Since $\mathbf{G}_i$ is a symmetric matrix (n x n), only n(n+1)/2 elements need to be stored. Our system uses a block diagonal matrix and the total size of one set of $\mathbf{k}_i$ and $\mathbf{G}_i$ is 1464 bytes. The small size enable us to save adaptation statistics for a large number of speaker clusters, which would not be possible with a model-based approach. In our current implementation, we provide space for 500 speaker clusters in a circular buffer so that the earliest cluster is erased when space for a new cluster is needed.

We use Viterbi alignment to align the utterance features to the SI model. The accumulation of $\mathbf{k}_i$ and $\mathbf{G}_i$ can be integrated within the Viterbi alignment process by re-arranging formulae (3) and (4):

$$\mathbf{G}_i = \sum_{\tau=1}^{T} \zeta(\tau)\zeta(\tau)^T \sum_{m=1}^{M} \left\{ \frac{1}{\sigma_i^{(m)2}} \right\} \gamma_m(\tau) \tag{6}$$

$$\mathbf{k}_i = \sum_{\tau=1}^{T} \zeta(\tau)^T \sum_{m=1}^{M} \left\{ \frac{\mu_i^{(m)}}{\sigma_i^{(m)2}} \right\} \gamma_m(\tau) \tag{7}$$

By moving the model loop inside the expression, the accumulation can be carried out during the Viterbi backtrace. Values in {} can be pre-computed.

The most computationally expensive process is the Gaussian computation required for $\gamma_m(\tau)$, as shown in formula (5). These same values are also required and thus calculated during the Viterbi alignment. We could save these values to avoid re-computing them. However, it is unknown which model ends up on the best path until the end of the Viterbi alignment is reached. Saving all the values for all active models during the search is impractical. To address this issue, we adopted two strategies:

1. Use a very tight pruning for the Viterbi alignment so that there are only a limited number of surviving theories.
2. Only use the top 5 Gaussian (M=5) components to approximate $\gamma_m(\tau)$, which requires storage of 5 floating-point numbers per surviving theory.

3.4. Online Speaker Adaptation Framework

Our LVCSR system is based on a 2-pass decoding strategy [7]. We designed the online speaker adaptation framework to integrate into both decoding passes.

Figure 1 illustrates the proposed general framework for integrating adaptation into one pass of the decoder.

For the first decoding pass, the only inputs are utterance features with a cluster label. The cluster label is used to retrieve the cluster statistics $\mathbf{k}_i$ and $\mathbf{G}_i$ from the "Cluster Statistics" database. A transformation matrix is then estimated from these statistics. The utterance features are transformed with the matrix and decoded. The decoding result is the output of this decoding pass and it is used as the transcript to accumulate adaptation statistics for the corresponding cluster. The new statistics replace the old ones in the database. No statistics are accumulated for the first utterance in the cluster, in which case the transformation matrix is set to the unity matrix and a new set of statistics is created in the database.

For the subsequent decoding pass, the process is virtually identical except that the recognition result from the preceding decoding pass is used as additional input, indicated as the shaded input data in **Figure 1**. These results are aligned to the SI model for the current decoding pass and adaptation statistics are accumulated for the cluster. In this case, even for the first utterance in the cluster statistics are available to estimate an adaptation transformation matrix.

The proposed framework does not require a re-decode of an utterance after adaptation. Additional latency is introduced by the Viterbi alignment, statistics accumulation, and transformation estimation. All three processes are very efficient and result in very low computational overhead, as shown in the experimental results.

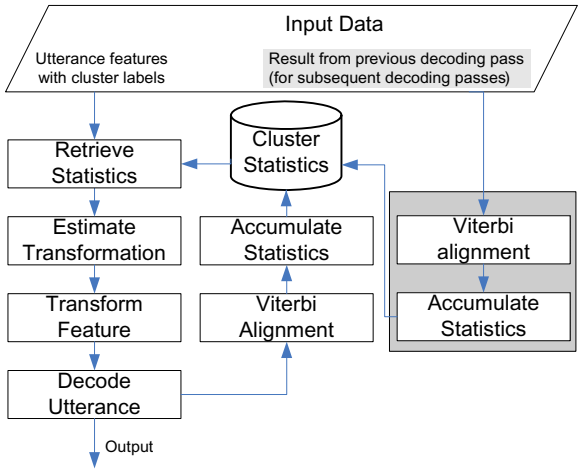

Figure 1. General framework for online speaker adaptation

4. Experimental Results

4.1. BBN Audio Indexer

All experiments are carried out using the BBN Audio Indexer System [8], a real-time end-to-end audio indexing engine that supports the languages: English, Chinese, and Arabic [9], [10]. The LVCSR engine in BBN Audio Indexer employs a 2-pass (forward and backward) decoding strategy, followed by a final nbest rescoring pass.

The forward pass is based on a fast-match algorithm described in [11], and makes use of a composite set bigram language model along with a 5-state non-crossword composite triphone Hidden Markov Model (HMM). The output of this pass consists of the most likely word ends per frame. This is used to restrict the search space in the backward pass.

The backward pass is a beam search that employs a trigram language model and non-crossword quinphone State Clustered Tied Mixture (SCTM) HMMs. The output of this pass is a list of nbest hypotheses for each utterance.

The nbest rescoring pass uses a crossword quinphone SCTM acoustic model to compute the acoustic likelihood score for each hypothesis in the nbest list. The top scoring hypothesis is the final recognition result.

Our speech feature consists of 14 Mel-Frequency Cepstral Coefficients(MFCC), first/second order cepstral derivatives, normalized energy, and first/second order of energy derivatives. The total feature dimension is 45. In the Chinese system, pitch and its first/second derivatives are also added which brings the dimension to 48.

4.2. Evaluation Condition

For performance evaluation, we used publicly available BN benchmark test data where available. The English test data consists of the Hub4 98/99 and RT-03 test set with a total of 10 hours of audio. The Chinese data consists of the Hub4 Non-English 97/98 evaluation set and 97 development set with a total of 3 hours. The Arabic test data was collected and transcribed in-house from 10 BN sources consisting of 15 hours of audio.

The real-time throughput reported in this section is measured on a dual-Xeon 2.0GHz PC with 2GB of RAM running Windows 2003 Standard Server.

4.3. Transformation Matrix Configuration

We used the 6-hour Hub4 98/99 English test set to determine the transformation matrix configuration from 4 choices:

1. Full matrix (45x45).
2. 2-Block-Diagonal (2-BD). One block contains the base cepstral coefficients. The other block contains energy and its derivatives.
3. 4-Block-Diagonal (4-BD). This configuration adds 2 blocks for the first and second cepstral derivatives to 2-BD.
4. 2-Block-Diagonal but adapting 4 Blocks (2-BD with 4-BD adapt). In this configuration, the transformation matrix is estimated with 2-BD and then applied to both base and derivative cepstral coefficients. This configuration is based on the assumption that the matrix estimated from the base coefficients should also be applicable to the derivatives since the derivative operation is a linear transformation,.

Table 1 shows the performance for these configurations.

Matrix configuration	WER	xRT
baseline, no adaptation	20.2	0.58
Full	19.1	1.12
2-BD	19.0	0.63
4-BD	18.4	0.64
2-BD with 4-BD adapt	18.6	0.63

Table 1. Comparison of matrix configurations on Hub4 98/99

All configurations improve the baseline performance. Full matrix is the slowest, which is expected due to the higher computational demand. However, it also resulted in the smallest WER gain. 4-BD performed superior to 2-BD and the 0.01xRT difference is negligible. 2-BD with 4-BD adapt is close to 4-BD in WER which validates the use of the base transformation for the derivatives. However, the saving in speed is not significant enough to justify the WER loss. 4-BD configuration is our final choice.

4.4. Offline vs. Online

To compare the performance of the online speaker adaptation to the offline mode, we evaluated both on identical speaker segmentation and clustering. Since offline adaptation uses more data for the transformation estimation, we expect it to outperform online adaptation. The performance numbers are shown in **Table 2**.

Mode	WER
Offline	18.3
Online	18.4

Table 2. Offline vs. Online, compared on Hub4 98/99

Surprisingly, the offline performance is only marginally better than the online performance.

4.5. System Performance

Based on the 4-BD matrix configuration, we evaluated the online speaker adaptation framework for the three languages. The results are presented in **Table 3** .

Language	Adaptation	WER	%gain	xRT
English	none	19.4	9	0.58
	online	17.6		0.64
Arabic	none	16.4	8	0.50
	online	15.1		0.53
Chinese	none	16.8	12	0.64
	online	14.8		0.73

Table 3. System performance with online speaker adaptation

Consistent improvements in WER are observed for all languages. The relative gain ranges from 8 to 12%. The cost in RT throughput is between 0.03xRT and 0.09xRT. Due to the added pitch features, the Chinese system requires one extra block in the block diagonal structure. This may explain why the Chinese system has the highest cost in real-time throughput.

5. Conclusions

We have developed an online speaker adaptation framework that provides an 8-12% relative WER gain to our real-time end-to-end BBN Audio Indexer System. The framework adds little overhead in latency and memory footprint. The additional cost to real-time throughput is less than 0.1xRT. The framework design is general so that it can be integrated easily into every decoding pass of a multi-pass decoder.

The new adaptation framework is currently utilized in BBN Audio Indexer Systems that are running 24x7 on live satellite feeds from English, Arabic, and Chinese broadcast news channels. The average system latency is around 50 seconds.

The framework can be extended in a number of future directions. The current 2-step speaker tracking approach can be abandoned in favor of an integrated approach where speaker boundaries and speaker clusters are jointly detected.

The small difference between offline and online results (**Table 2**) implies that there might be a condition beyond which additional data would no longer improve adaptation performance. The system can be modified to skip the accumulation of statistics for a cluster when that condition is reached.

State-of-the-art speech recognition systems frequently fail during field operation due to unexpected field conditions. We consider our effort as a first step towards a practical continuous learning system design, where the system continuously adapts itself by enriching its knowledge about the field environment. In this paper, the only knowledge source we use is speaker identity. In the future, the same framework could be applied to other environment conditions.

6. Acknowledgement

This work was supported in part by the Defense Advanced Research Projects Agency under contract N66001-00-C-8008. The views and findings contained in this material are those of the authors and do not necessarily reflect the position or policy of the Government and no official endorsement should be inferred.

7. References

[1] V. V. Digalakis, “Online Adaptation of Hidden Markov Models Using Incremental Estimation Algorithms,” *IEEE Transactions on Speech and Audio Processing,* Vol. 7, No. 3, pp253-261, May 1999

[2] C. Mokbel, “Online Adaptation of HMMs to Real-Life Conditions: A Unified Framework,” *IEEE Transactions on Speech and Audio Processing,* Vol. 9, No 4., pp342-357, May 2001

[3] Z. Zhang, et al, “On-Line Incremental Speaker Adaptation with Automatic Speaker Change Detection,” *ICASSP’2000*, Vol 2, pp961-964, 5-9 June, 2000

[4] D. Liu, F. Kubala, “Fast Speaker Change Detection for Broadcast News Transcription and Indexing,” *EUROSPEECH’99*, Budapest, Hungary, Volume 3, Page 1031-1034, September 5-9, 1999

[5] D. Liu, F. Kubala, "Online Speaker Clustering," *ICASSP'04*, Montreal, Canada, page 333-336, May 17 - 21, 2004

[6] Z. Wang, et al, “Comparison of Acoustic Model Adaptation Techniques on Non-Native Speech,” *ICASSP’03*, vol-I, pp540-543, Hong Kong, 2003

[7] M. Gales, “Maximum Likelihood Linear Transformations for HMM-Based Speech Recognition,” *Technical Report CUED/F-INFENG/TR 291*, Cambridge University, May 1997

[8] F. Kubala, et al, “Integrated Technologies for Indexing Spoken Language,” *Communications of the ACM*, No. 2, 48–56 (February 2000)

[9] D. Liu, et al, “Real-Time Rich-Content Transcription of Chinese Broadcast News,” *ICSLP’2002*, pp 1981-1984, Denver, CO, September, 2002

[10] J. Billa, et al, “Audio Indexing of Arabic Broadcast News,” *ICASSP 2002*, Orlando, FL, May 2002

[11] L. Nguyen, R. Schwartz, “Efficient 2-Pass NBest Decoder,” *Eurospeech’97*, Rhodes, Greece, September 1997, pp 167-170