Towards High Performance LVCSR in Speech-to-Speech Translation System on Smart Phones

Jian Xue, Xiaodong Cui, Gregg Daggett, Etienne Marcheret, and Bowen Zhou

IBM T.J. Watson Research Center, Yorktown Heights, NY 10598, USA

Emails: {jxue,cuix,gdaggett,etiennem,zhou}@us.ibm.com

ABSTRACT

This paper presents the endeavors to improve the performance of large vocabulary continuous speech recognition (LVCSR) in speech-to-speech translation system on smart phones. A variety of techniques towards high LVCSR performance are investigated to achieve high accuracy and low latency given constrained resources. This includes one-pass streaming mode decoding for minimum latency, acoustic modeling with full-covariance based on bootstrap and model restructuring for improving recognition accuracy with limited training data; quantized discriminative feature space transforms and quantized Gaussian mixture model to reduce memory usage with negligible degradation on recognition accuracy. Some speed optimization methods are also discussed to increase the recognition speed. The proposed techniques evaluated on the DARPA Transtac datasets will be shown to give good overall performance under the constraints of both CPU and memory on smart phones.

Index Terms: speech recognition, speech-to-speech translation, one-pass streaming, bootstrap, quantized discriminative feature space transforms

1. INTRODUCTION

In the past several years, the performance of LVCSR systems has been improved significantly by applying advanced and effective algorithms such as feature based Minimum Phone Error (fMPE) [1], feature based Maximum Mutual Information (fMMI) [2], multi-pass with post-processing decoding method [3], etc. While these approaches have achieved superior performance, they typically require more powerful platforms such as servers or laptop computers, due to the high complexity of the system. Recently, more and more efforts have been spent on speech technologies (e.g. speech recognition, speech translation, voice search) on smart phones which have emerged as an integrated communication, data processing and entertainment platform. For example, the DARPA Transtac Program used the Google/Android Nexus One smart phone as the platform for the 2010 Transtac S2S evaluations. Although the hardware capability of smart phones has increased dramatically, it still can not meet our requirements to some extent. In the DARPA Transtac program, a two way speech-to-speech (S2S) translation system has to be deployed on a single device, which includes two ASR components (English and foreign languages), two text-to-speech (TTS) components (English and foreign languages), and two machine translation (MT) components (Enligh-to-foreign and foreign-to-English). CPU and memory limitations of the smart phone platform restrict the footprint of individual components, as well as the performance. Furthermore, the ASR accuracy is often hindered by limited availability of training data also, especially for low-resourced languages such as Farsi, Dari and Pashto where the availability of high quality transcribed speech is limited due to the difficulty of extensive data collection and expensive labor for audio transcription. This paper investigates techniques for high ASR accuracy and low latency with small footprint on smart phones.

This material is based upon work supported by the DARPA Transtac project.

The rest of the paper is organized as the following. Section 2 briefly introduces the infrastructure of the IBM S2S translation system used in the 2010 DARPA Transtac evaluations. A variety of techniques to achieve high LVCSR performance on smart phones are described in section 3, which includes one-pass streaming mode decoding scheme, acoustic modeling with full-covariance based on bootstrap and model restructuring, quantized discriminative feature space transforms and GMMs, as well as the optimization work for recognition speedup. Experimental results are provided in section 4 followed by a summary in section 5.

2. S2S TRANSLATION SYSTEM INFRASTRUCTURE

The infrastructure of the IBM S2S translation system with the major components is shown in Fig. 1. In this client-server architecture, the ASR, MT and TTS components are located in the background as services. Each component provides two types of services as shown in the figure. Users control the system through client application. The client communicates with each component through socket.

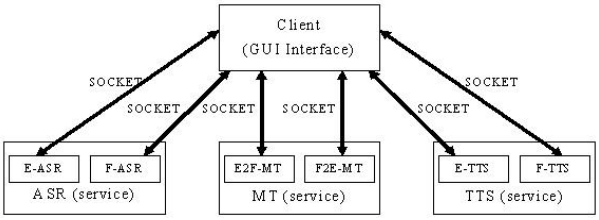

Fig. 1. *IBM S2S translation system on smart phones.*

Google/Android Nexus One (N1) was used as the platform in the 2010 DARPA Transtac S2S evaluations. N1 has 1GHz ARM Snapdragon CPU and 512 MB RAM. Excluding the memory required as the fundamental resources of the operating system (OS), the total memory available for running the S2S translation system is around 300MB. Moreover, the MT and TTS components need significant amount of memory to achieve comparable performance with the system on laptop. So the memory usage of two-way ASR components is limited to no more than 150MB. Given the above constraints on both computational power and memory, numerous techniques have been explored to strive for the best overall performance on the smart phones, which will be elaborated in Section 3.

3. TECHNIQUES FOR HIGH PERFORMANCE ASR

3.1. One-pass streaming mode decoding

Decoding latency is an important factor when designing real-time ASR for S2S systems, which is the overall processing time spent after the whole speech utterance is received. In the IBM S2S system, the MT and TTS functions cannot be started until the corresponding ASR processing is done. So the latency of ASR will highly determine the usability of the S2S system. In real-time ASR, the decoding latency not only depends on the decoding speed, but also on the style of the decoding procedure. To reduce latency and also increase speed, a one-pass streaming mode decoding scheme is employed which achieves a very short decoding latency by limiting the length of look-ahead.

The ultimate feature sets are computed through multiple feature layers. A final feature set X representing an utterance with input waveform set $Y = \{y_1, y_2, \cdots, y_\mathrm{T}\}$ can be depicted as:

$$X|_1^\tau = \Phi(Y|_1^\mathrm{T}) = \Phi_{\mathrm{fMLLR}}\{\Phi_{\mathrm{fMMI}}\{\Phi_{\mathrm{LDA}}\{\Phi_{\mathrm{CMN}}\{\Phi_{\mathrm{PLP}}(Y|_1^\mathrm{T})\}\}\}\} \quad (1)$$

In streaming mode decoding [4], feature set $\mathbf{X}$ is computed incrementally in the time domain. According to Eq.1, the final feature set $X|_1^t$ for a chunk of speech audio is computed as:

$$X|_1^t = \Phi_{\mathrm{L}_5}(Y_{\mathrm{L}_4}|_1^{t+\mathrm{W}_{\mathrm{L}_5}}) \quad (2)$$

$$Y_{\mathrm{L}_i}|_1^\mathrm{T} = \Phi_{\mathrm{L}_i}(Y_{\mathrm{L}_{i-1}}|_1^{t+\mathrm{W}_{\mathrm{L}_i}}), \quad i = 1, \cdots, 4 \quad (3)$$

where $Y_0 = Y$, $\mathrm{L}_1 = \mathrm{PLP}$, $\mathrm{L}_2 = \mathrm{CMN}$, $\mathrm{L}_3 = \mathrm{LDA}$, $\mathrm{L}_4 = \mathrm{fMMI}$, $\mathrm{L}_5 = \mathrm{fMLLR}$, and $\mathrm{W}_{\mathrm{L}_i}(i = 1, \cdots, 5)$, are feature-layer dependent look-ahead decoding parameters. As a result, Eq.2 can be re-written as

$$X|_1^t = \Phi(Y|_1^{t+max(\mathrm{W}_{\mathrm{PLP}}, \mathrm{W}_{\mathrm{CMN}}, \mathrm{W}_{\mathrm{LDA}}, \mathrm{W}_{\mathrm{fMMI}}, \mathrm{W}_{\mathrm{fMLLR}})}) \quad (4)$$

3.2. Full-covariance based bootstrap and model restructuring

To obtain high performance in acoustic modeling with limited training data (which is the case for DARPA Transtac project on low-resourced languages such as Dari and Pashto), full-covariance based bootstrap and model restructuring is employed [5][6], which includes two major steps: model aggregation with bootstrap and model restructuring.

3.2.1. Model Aggregation with Bootstrap

Suppose S is the original training data set. N subsets of data, $S_1, S_2, \cdots, S_N$, are generated by resampling from S without replacement. An HMM, denoted as $\lambda_{\mathrm{bs},i}$, is estimated from each individual subset S_i. All HMMS $\lambda_{\mathrm{bs},i}$ share the same LDA, global semi-tied covariance (STC) and decision tree which are built on the whole ensemble of resampled subsets. So the Gaussian likelihood of feature x on state s is calculated as

$$\begin{aligned} f_s(x) &= \frac{1}{N}\sum_{i=1}^{N}\sum_{k_{is}=1}^{K_{is}} c_{isk}\mathcal{N}(x;\mu_{isk},\Sigma_{isk}) \\ &= \sum_{i=1}^{N}\sum_{k_{is}=1}^{K_{is}} \omega_{isk}\mathcal{N}(x;\mu_{isk},\Sigma_{isk}) \end{aligned} \quad (5)$$

with $\omega_{isk} = c_{isk}/N$. We use full covariance here since full covariance contains richer structural information than diagonal covariance. The aggregated model with full covariance is shown to give significantly better performance over single systems built from the original training data. However, this improved performance comes at a cost of substantially larger model size. Therefore, model restructuring needs to be employed to scale down the model size.

3.2.2. Model Restructuring

Fig. 2 illustrates the process of model restructuring. Given the aggregated model λ_b with full covariance, Gaussian clustering is first performed to reduce Gaussian components to a reasonable number which is followed by a model refinement to minimize certain "distance" between the down-scaled GMM and the original aggregated large GMM. These two steps are carried out in the full covariance space. Then the down-scaled GMM in full covariance is converted to one in diagonal covariance. Lastly, final model λ_r is obtained by another model refinement conducted in the diagonal covariance space. For the details of model restructuring refer to in [6].

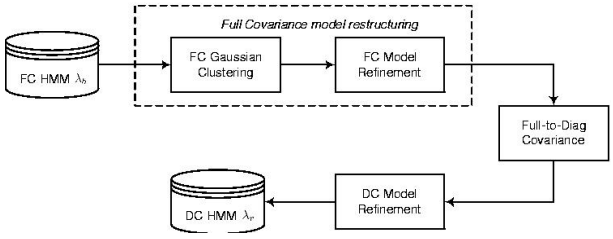

Fig. 2. *Model restructuring of the aggregated model with full covariance (FC) and conversion to diagonal covariance (DC).*

3.3. Quantization in acoustic modeling

Discriminative training of the feature space achieved superior improvement in recognition accuracy. However, the price of these gains is a parameter space consisting of millions of parameters which makes it difficult to deploy on smart phones. Also we cannot afford a large number of GMMs due to the memory limitation. To reduce the memory requirements while keeping the gains of the recognition accuracy, quantization methods were investigated.

3.3.1. Quantized discriminative feature space transforms

The fMMI process can be described by two fundamental stages. The first stage, level 1, relies on a set of Gaussians $\mathcal{G}$ to convert an input d-dimensional feature vector x_t to *offset* features

$$\mathbf{o}(t,g,i) = \begin{cases} \gamma_g \frac{\left(x_t^{(i)} - \mu_g^{(i)}\right)}{\sigma_g^{(i)}} & \text{if } i \le d \\ 5\gamma_g & \text{if } i = d+1 \end{cases} \quad (6)$$

where t denotes time, and i denotes offset dimension. γ_g is the posterior probability of $g \in \mathcal{G}$ given x_t. The set $\mathcal{G}$, of size G, is arrived at by clustering the Gaussians of the original acoustic model.

In general $\mathbf{o}(t,g,i)$ contains $(d+1)G$ elements for each time t. For computational efficiency all γ_g below a threshold γ_{cut} are set to 0 resulting in a sparse $\mathbf{o}(t,g,i)$.

The offset features are operated on by a level 1 transform $M^1(g,i,j,k)$

$$\begin{aligned} b(t,j,k) &= \sum_{g,i} M^1(g,i,j,k)\mathbf{o}(t,g,i) \\ &= \sum_{g:\gamma_g>\gamma_{\mathrm{cut}}}\sum_{i} M^1(g,i,j,k)\mathbf{o}(t,g,i). \end{aligned} \quad (7)$$

where M^1 is parameterized by Gaussian $g \in \mathcal{G}$, offset dimension $i \in \{1, \ldots, d+1\}$, an *outer-context* $j \in \{1, \ldots, 2J+1\}$ and final output dimension $k \in \{1 \,..\, d\}$.

The next stage of fMMI, level 2, takes as input $b(t+\tau, j, k)$ for $\tau \in \{-\Lambda, \ldots, \Lambda\}$. It computes its output as

$$\delta(t,k) = \sum_j \sum_\tau M^2(j,k,\tau+\Lambda+1) b(t+\tau, j, k) \qquad (8)$$

The output of level 2, $\delta(t,k)$, is added to $x_t(k)$ to compute the fMMI features.

In the setup discussed in this paper, $G = 512$, $d = 40$, $J = 4$, and $\Lambda = 8$. This results in M^1 with $512 * 41 * 40 * 9 = 7557120$ parameters. The posterior threshold γ_{cut} is typically 0.1, resulting in a small number of active Gaussians per x_t. For each active Gaussian, level 1 requires $41 * 40 * 9 = 14760$ floating point operations. At level 2, M^2 contains $9 * 40 * (2 * 8 + 1) = 6120$ parameters, and computation of $\delta(t,k)$ at level 2 requires 6120 floating point operations.

As seen above, the level 1 fMMI process dominates in the amount of CPU and memory used. For the example given here, 7.55 million M^1 parameters use 30.2MB of memory, almost double the memory of 40 dimensional 50K diagonal gaussians, which is around 16MB. Therefore it is clear some form of compression is needed for the M^1 transform in resource constrained environments that has minimal negative impact on the system accuracy.

To quantize level 1 transform M^1, we adopted the strategy of quantizing blocks of parameters using separate quantization tables. Once the blocks were decided, we chose number of quantization levels to use for each block. The quantization values were then initialized and each parameter was assigned to a quantization value.

Since the fMMI pertubation $\delta(t,k)$ shown in equation 8 decouples across feature vector dimension k, we choose as the objective function to minimize:

$$E = \sum_{t,k} \left(\delta(t,k) - \delta^Q(t,k) \right)^2 \qquad (9)$$

where $\delta^Q(t,k)$ denote the feature perturbation obtained using the quantized level 1 transform M^{1Q}.

Using partition indicators $I_p(g,i,j,k)$, and quantization table $\mathbf{q} = (q_1, \ldots, q_n)^T$ $M^{1Q}(g,i,j,k)$ can be written as

$$M^{1Q}(g,i,j,k) = \sum_p q_p I_p(g,i,j,k). \qquad (10)$$

To ensure that $M^{1Q}(g,i,j,k)$ is equal to one of the quantization values in $\mathbf{q}$, we impose the additional constraint that for each (g,i,j,k) only one of $I_p(g,i,j,k)$ can be 1, i.e. the indicators form a partition of the parameter indices. Replacing $M^1(g,i,j,k)$ in equation 7 by the quantized $M^{1Q}(g,i,j,k)$ of equation 10 results in a quadratic expression for E of equation 9. The natural decoupling that occurs across dimensions allows us to optimize E by dimension ($E(k)$), which is straightforward to do because of the quadratic form. The complete details can be found in [7].

To initialize the partition indicators $I_p(g,i,j,k)$, and the quantization values q_p shown in equation 10, parameters corresponding to each dimension k in $M^1(g,i,j,k)$ were quantized separately using their own quantization table. The K-means algorithm [9] was used to determine the quantization levels and indicators. We refer to this method as `DimK` and the quantization values resulting from the optimization of equation 9 will be referred to as learned. Partition indicators were not updated in this work, details of that additional optimization are available in [7].

3.3.2. Quantized GMMs

Suppose the feature space is d-dimensional. These dimensions are divided into subsets of dimensions, referred to as bands. The original Gaussians in each band are quantized into a certain number of Gaussian components (we set the number of Gaussian components in each band be less than 256, so that the index of each Gaussian can be stored in one byte). For each feature vector, we first calculated the log likelihoods of all the Gaussian components in each band, which are stored in a lookup table. Then the log likelihood for the original Gaussian is obtained as the sum of the log likelihood values of the corresponding Gaussian components in the pre-stored lookup table. Refer to in [10] for the details.

3.4. Speed optimization

3.4.1. Parallelization for Decoding Speedup

One approach proposed to improve the ASR decoding speed on smart phones is to parallelize certain portions of the decoding computation (such as the multi-dimension Gaussian computation) for running on CPU's which support a SIMD (Single Instruction, Multiple Data) architecture. For example the ARM Cortex(tm) series of processors (which are used in the Apple iPhone and many other popular smart phone devices) support the NEON(tm) 4-way SIMD floating-point computation pipeline, allowing substantial speedup of compute-intensive tasks.

Since a diagonal covariance matrix is used in the Gaussian density function, the Gaussian computation can be parallized by using SIMD processor instructions to compute several dimensions simultaneously. For ARM NEON-enabled CPU's, it achieved a 2× speedup in the Gaussian likelihood computation when computing four dimensions in parallel. This resulted in an ASR decoding speedup of approximately 20% when running on a popular smart phone which supports these SIMD instructions.

Parallelization has also improved overall decoding speed by an additional 6% when other compute-intensive sections of the ASR engine are implemented using SIMD instructions, therefore resulting in an overall speedup of approximately 26% relative to a non-parallel engine implementation.

3.4.2. Sequential Online Speaker Adaptation

Online speaker adaptation is a crucial component in our ASR components deployed in S2S translation system. To achieve low latency of the system, speaker adaptation has to be carried out on the fly. For computational simplicity given the latency requirement, online incremental unsupervised fMLLR is used to dynamically collect acoustic statistics from the speaker and update the acoustic models as the system being used [8].

The acoustic models are updated after finishing recognizing each utterance, if the amount of collected acoustic statistics is more than a predefined threshold. To achieve minimal system latency, model update is performed when the system is playing TTS after finishing MT and TTS.

4. EXPERIMENTAL RESULTS

The experiments were conducted on the datasets of the DARPA Transtac program. Experimental results on Dari and English are presented. For Dari there are 135 hours of training data and 10 hours of test data, and for English there are about 270 hours training data and 10 hours of test data. The feature space is constructed by splicing

9 frames of 24 dimensional PLP features and projecting down to a 40 dimensional space via LDA followed by a global STC. Context-dependent quinphone states are tied by a decision tree. For Dari we use trigram language models with 1.8M n-grams, and for English we use trigram language models with 3.6M n-grams. The dictionary size is 60K words for English, and 49K for Dari.

4.1. Recognition Accuracy

Table 1 shows the word error rate (WER) of English for two quantized fMMI configurations (4 quantization levels (lvl) and 2 lvl) along with the unquantized fMMI baseline system. The 4 lvl and 2 lvl quantizations translate into 2bit and 1bit per fMMI output dimension respectively. The required memory for the 7.5M parameters as detailed in section 3.3, drops from 30M to 1.89M for the 2 bit condition and 0.94M for 1 bit. It is surprising here that 4 level k-means outperforms the baseline, this could be an fMMI overtraining condition and mismatch at test time. For deployment we choose the 4 level learned, as we trust this more than the k-means result and the 2 level was more compression than we needed. The baseline acoustic model uses 3K HMM states and 50K diagonal Gaussians, trained without the bootstrap method since there was not a training data sparsity condition.

model	fMMI Mem (MB)	WER
baseline	30.2	18.8%
`DimK` 4 lvl	1.89	18.6%
+learned	1.89	18.9%
`DimK` 2 lvl	0.94	18.9%
+learned	0.94	18.8%

Table 1. WERs of English ASR in Quantized FMMI `DimK` and learned transformation

Table 2 gives the WER results for Dari. Bootstrap and model restructuring are applied to train Maximum Likelihood (ML) models, in which the original training data is bootstrapped into 10 subsets with each subset covering 70% of the original training set. On top of that discriminative training is applied. From the table we observe that the bootstrap method on ML level (BS_ma_full(ML)) improves the accuracy 4% absolute, with ten times the number of Gaussians. Gaussian clustering and model refinement (BS_cr_full(ML)) reduce the improvement to 3.3% with comparable Gaussian numbers. Full-to-diagonal conversion and model refinement in the diagonal covariance space reduce the improvement to 3.0% with comparable model size. After discriminative training we can see that the full-covariance based bootstrap and model restructuring methods improve the accuracy by 1.7% absolute, which is smaller than the gain on the ML level. After quantization we reduce the memory usage while keeping the same performance.

model	model size (states/Gaussians)	WER
baseline(ML)	3K/50K	46.2%
BS_ma_full(ML)	5K/540K	42.2%
BS_cr_full(ML)	5K/60K	42.9%
BS_f2d_rf_diag(ML)	5K/60K	43.2%
baseline(fMMI+BMMI)	3K/50K	38.5%
BS(fMMI+BMMI)	5K/60K	36.8%
BS+K-means(fMMI+BMMI)	5K/60K	36.8%
BS+K-learned(fMMI+BMMI)	5K/60K	36.8%

Table 2. WERs of Dari systems of baseline and bootstrap and quantized fMMI

4.2. Speed and memory usage

After quantization in fMMI and GMMs, the total memory usage for ASR on both directions at run time reduces from 220MB to 130MB, among which quantized fMMI saved around 60MB, and quantized GMMs saved around 30MB. In streaming mode decoding the chunk size is set to be 100 ms, and the overall speed is around $0.9\times$RT. So the total ASR latency is around 100 ms.

5. SUMMARY

In this work we have presented several methods to improve the LVCSR performance on smart phones. One-pass streaming mode decoding with multiple feature layers is used to achieve minimum decoding latency. Full-covariance based bootstrap and model restructuring methods are used to train robust acoustic models with limited acoustic data. Quantization methods reduce memory usage with negligible accuracy loss. The decoding speed is further optimized using parallelization optimization. Compared to the system on laptop computers, the experimental results show that similar recognition performance can be obtained with the small footprint system on smart phones.

6. REFERENCES

[1] D. Povey, B. Kingsbury, L. Mangu, G. Saon, H. Soltau, and G. Zweig, "fMPE: Discriminatively trained features for speech recognition," in *ICASSP*, 2005, pp. I961–I964.

[2] D. Povey, D. Kanevsky, B. Kingsbury, B. Ramabhadran, G. Saon, and K. Visweswariah, "Boosted MMI for model and feature-space discriminative training," in *ICASSP*, 2008, pp. 4057–4060.

[3] S. Matsoukas, R. Prasad, S. Laxminarayan, B. Xiang, L. Nguyen, and R. Schwartz, "The 2004 BBN 1xRT recognition systems for english broadcast news and conversational telephone speech," in *Interspeech*, 2005, pp. 1641–1644.

[4] L. Gu, J. Xue, X. Cui, and Y. Gao, "High-performance low-latency speech recognition via multi-layered feature streaming and fast gaussian computation," in *Interspeech*, 2008, pp. 2098–2011.

[5] X. Cui, J. Xue, P. L. Dognin, U. V. Chaudhari, and B. Zhou, "Acoustic modeling with bootstrap and restructuring for low-resourced languages," in *Interspeech*, 2010, pp. 2974–2977.

[6] X. Cui, X. Chen, J. Xue, P.A. Olsen, J.R. Hershey, and B. Zhou, "Acoustic modeling with bootstrap and restructuring based on full covariance," Submitted to *Interspeech* 2011.

[7] E. Marcheret, V. Goel, and P. Olsen, "Optimal quantization and bit allocation for compressing large discriminative feature space transforms," in *ASRU*, 2009, pp. 64–69.

[8] Y. Li, H. Erdogan, Y. Gao, and E. Marcheret, "Incremental online feature space MLLR adaptation for telephony speech recognition," in *ICSLP*, 2002, pp. 1417–1420.

[9] R. O. Duda, P. E. Hart, and D. G. Stork "Pattern Classification, Second Edition", John Wiley & Sons, Inc., 2001.

[10] M. Novak, R. Gopinath, and J. Sedivy, "Efficient hierarchical labeler algorithm for Gaussian likelihoods computation in resource constrained speech recognition systems," `http://www.research.ibm.com/people/r/rameshg/novak-icassp2002.ps`