Recent Advances of IBM's Handheld Speech Translation System

Weizhong Zhu, Bowen Zhou, Charles Prosser, Pavel Krbec and Yuqing Gao

IBM T. J. Watson Research Center
Yorktown Heights, New York 10598
{williamz,zhou,prosser,pavel_krbec,yuqing}@us.ibm.com

Abstract

Recent Advances in the processing capabilities of Personal Digital Assistants (PDAs) have enabled the implementation of an end-to-end speech translation system on these devices. We have presented a bi-directional speech-to-speech (English and Chinese) translation system, which is hosted on a PDA running embedded Linux. Our Multilingual Automatic Speech-to-Speech Translator (MASTOR) system includes an HMM-based large vocabulary continuous speech recognizer using statistical n-grams, a translation module, and a multi-language speech synthesis system. This paper describes our recent efforts to develop a bi-directional English and colloquial Arabic speech-to-speech system on a device running the popular Windows-CE operating system. We have created completely new translation modules and introduced a fast adaptation scheme for a new speaker or environment. In addition, the componentization of system modules speeds up the development of a new speech-to-speech system.

Index Terms: speech recognition, machine translation, speech synthesis, adaptation, PDA

1. Introduction

In recent years, there have been significant efforts to develop reliable and satisfactory automatic speech-to-speech translation systems, which are typically available on more powerful resources such as desktop servers or laptop computers. However, because such devices are not compact, they are not convenient for mobile applications. This limits the usefulness of this form of translation technology. Many circumstances where translation is required can only be effectively aided by truly mobile devices such as a Personal Digital Assistant (PDA).

On the one hand, automatic speech-to-speech translation is a highly complex task. A large amount of computation is required to achieve reliable translation performance. Memory and storage requirements, and the audio input and output requirements all tax current systems to their limits. On the other hand, the development of increasingly powerful mobile devices is reaching a level that is comparable to the power of desktop systems of only a few years ago. In order to bridge the gap between the requirements of contemporary translation systems and the current mobile computing platforms, we have employed a number of optimizations, significantly enhancing the accessibility of our automatic speech translation technology [1]. In this paper, we further extend our previous work, and present a complete bi-directional English and colloquial Arabic speech-to-speech translation system that is deployed on an off-the-shelf PDA with the popular Windows-CE platform.

2. System Overview

2.1. System Architecture

Our system employs the same architecture as its desktop counterpart, the MASTOR system [2]. Specifically, the system consists of a Large Vocabulary Continuous Speech Recognizer (LVCSR) that operates in real-time or near-real time to recognize input utterances; a fast translation module to translate the recognized text from the source language into text in the target language; and a multi-language speech synthesizer, to convert the translated text into audio output in the target language. The system GUI is designed to let the user check both recognition and translation results, and to allow the user to re-play the output. Logging of results from the recognition and translation modules, as well as system configuration are implemented in the system.

2.2. Hardware Specifications

To demonstrate the feasibility of building a system with our speech translation architecture on a standard PDA, we have built our system on two target hardware platforms. One is a custom designed PDA, (referred to as the P2 below). It is equipped with an Intel 400 MHz XScale PXA 255 processor. The system has 256MB of RAM and an SD card for additional storage. The other platform is the HP iPaq PDA model H2750, a popular and widely available device. The processor shipped with this product is Intel's XScale PXA 270 running at a frequency of 624 MHz. The system has 128 MB of RAM and an 82 MB iPAQ File Store system. It is also equipped with an SD card for additional storage. Both systems have a built-in microphone for speech input and an integrated speaker for audio output. The P2 has been ruggedized for outdoor use. The original P2 was equipped with software for one-way, fixed phrase translation.

3. System Components

3.1. System Interface

Figure 1 shows the MASTOR user interface for the handheld device. On the top, there are two radio buttons showing the status of the microphone. In the middle, there are two edit controls, one for displaying recognition results, the other for displaying translation results. The translation button is for initiating the translation. The play output button is for playing or re-playing the audio output. There is one *plus* and one *minus* button which let the user change the volume level of speech

Figure 1 *MASTOR system user interface.*

output. There are two radio buttons at the bottom showing the current speech translation direction. Along with the direction radio buttons, there is also an indicator which shows the state of the adaptation function in the Automatic Speech Recognition (ASR) engines. The action of turning on or off this adaptation function is in the system menu.

There two switch buttons and one toggle button on this specially designed PDA. We utilize the switch buttons to turn on or off the microphone for both languages. We use the toggle button for translation and playing output, as well as increasing or decreasing the playback volume. Therefore, the system on this PDA is stylus free, making it possible to operate with one hand.

3.2. LVCSR on a PDA

The recognition module developed for our mobile speech translation system is an HMM-based LVCSR engine using statistical n-grams. Unlike most grammar-based embedded speech recognition systems, our system has the advantage of large vocabulary coverage. Moreover, it has the flexibility to switch to new application domains, which is typically only found on desktop-based systems. To accomplish this, IBM's large vocabulary speech recognition engine, as featured in the popular ViaVoice dictation product, was ported to the XScale processor architecture.

3.2.1. Acoustic and Language models

The English acoustic model uses an alphabet of 52 phones. Each phone is modeled with a 3-state left-to-right hidden Markov model (HMM). This system has approximately 3,500 context-dependent states modeled using 42K Gaussian distributions and trained using 40 dimensional features. The context-dependent states are generated using a decision-tree classifier. The colloquial Arabic acoustic model uses about 30 grapheme phones that essentially correspond to letters in the Arabic alphabet, not including any diacritics such as short vowels. The colloquial Arabic HMM structure is the same as that of the English model. The colloquial Arabic acoustic model is also built using 40 dimensional features. It has 28K Gaussian distributions. Both models are trained using discriminative training [3].

A statistical trigram language model is built for both the English and colloquial Arabic languages recognized in this speech translation system. The English language model is built using a corpus of 6.4M words. This corpus is split as training and holdout sets with 5.7M and 0.64M words, respectively. The vocabulary size is about 30K. The language model is smoothed using a deleted interpolation technique. Due to memory limitations of the P2 device, the language model is further pruned with bigram and trigram thresholds of 1 and 2, respectively. The size of the English language model is about 7MB. In Arabic, words can take prefixes and suffixes to generate new words that are semantically related to the root form of the word (stem). As a result, the vocabulary size in modern standard Arabic as well as dialectal Arabic can become very large even for specific domains. For colloquial Arabic, we used a corpus of 3.3M words that is again split as training and holdout sets with 2.9M and 0.32M words, respectively. The vocabulary size for this corpus is about 98K, which is too large to be used on the P2 system due to the CPU and memory limitations on the device. Aggressive pruning of both the vocabulary and the counts of the language model would be required. Instead, we built the language model on morphologically tokenized data. Applying the morphological analysis, we split some of the words into prefix+stem+suffix, prefix+stem, or stem+suffix forms. We refer the reader to [4] to learn more about the morphological tokenization algorithm. Morphological analysis reduced the vocabulary size to 58K without sacrificing coverage. Nevertheless even this was too large to be used in the P2 device. Next, we eliminated singletons from the vocabulary, which reduced the vocabulary further down to 37K, and applied cutoff thresholds to the bigram and trigram counts in the language model. The size of the final language model is about 9MB.

3.2.2. Fast incremental adaptation

Adaptation to a new speaker or environment is becoming very important in embedded speech recognition as these systems are deployed in unpredictable real world situations. Feature space Maximum Likelihood Regression (fMLLR) has proven to be especially effective for this purpose, particularly when used for incremental unsupervised adaptation [5]. Unfortunately the standard implementation used by most authors requires unacceptable CPU power for embedded speech recognition systems. The CPU requirements can, to a degree, be lowered by using the block diagonal transformation matrix, but we will show that there are other problematic issues with the standard approach later.

We have decided to use the stochastic gradient descent approach. It has been successfully implemented in IBM's Embedded ViaVoice (EVV), where we face the fundamental problems of embedded systems: limited CPU performance, slow memory, no floating point unit. Adaptation is implemented through a feature space transform of the form $\mathcal{O}' = \mathcal{A}\mathcal{O} + \mathcal{B}$, where $\mathcal{O}$ are the speech frames, $\mathcal{A}$ is the transformation, and $\mathcal{B}$ the bias. The total amount of parameters to estimate is only $n(n+1)$, where n is the dimension of the feature vector. The adaptation is thus effective even with just a few seconds of data. One of the main challenges we face when deploying fMLLR on embedded platforms is integerization. The classical approach used in [5] requires the need to compute the inverse of $\mathcal{A}$. The inverse is usually performed using the Choleski decomposition algorithm. The implementation in integer arithmetic is fast, but unfortunately very sensitive to numerical errors (due to the necessary scaling and rounding), and can end up with completely wrong eigenvalues. This Choleski decomposition

Table 1. *Comparison of word error rate(%) with and without proposed adaptation method.*

Speaker	No adaptation	Adaptation	R.I.
asa	4.37	3.96	9.38
bxb	5.74	4.37	23.87
djl	5.60	5.05	9.82
lps	12.57	12.16	3.26
wja	2.46	1.91	22.36
Ave	6.15%	5.49%	10.73%

break down can be detected and an extra fail-safe mechanism usually takes care of resetting the transform in this case. In our solution we use the stochastic gradient descent approach which avoids the computation of the inverse [6] and thus the eigenvalue related problems do not exist.

3.2.3. Experiment results for speaker adaptation

Experiments are designed to measure the effectiveness of the proposed adaptive method. Experiments were conducted with 5 speakers, each with 150 English sentences. The comparison results of word error rate are shown in table 1. The proposed adaptive method works well for all 5 speakers. On average, the relative improvement of word error rate is 10.73%.

We also measured CPU usage with adaptation enabled and disabled. On average, the proposed stochastic gradient descent method only increases CPU usage by about 15%. Fortunately, most of this extra usage occurs at the end of recognition, after the user has been presented with recognition results, so it does not affect real time recognition performance.

3.3. Translation Module

Recently, Zhou et al. [7] introduce a novel phrase-based translation framework, which we refer to as a Statistical Integrated Phrase Lattice (SIPL). More details of this work will be presented elsewhere [7], and we provide some overview of this technique here for its application on PDA devices.

3.3.1. SIPL for Translation

We statically construct a single optimized Weighted Finite State Transducer (WFST), encoding the entire translation model. In addition, we describe a Viterbi decoder that can combine the translation model and language model FST's with the input lattice extremely efficiently. This results in translation speeds of hundreds of words per second on a PDA device, while using less than 20 MB runtime memory.

The phrase-based translation task can be framed as finding the best path in the following FSM, $\mathcal{S} = \mathcal{I} \circ \mathcal{H}$, where $\mathcal{I}$ represents the source sentence with possible reordering, and,

$$\mathcal{H} = \mathcal{P} \circ \mathcal{T} \circ \mathcal{W} \circ \mathcal{L} \qquad (1)$$

here $\mathcal{P}$, $\mathcal{T}$, $\mathcal{W}$, and $\mathcal{L}$ refer to the transducers of source language segmentation, the phrase translation, the target language phrase-to-word, and the target language model, respectively.

To minimize the amount of computation required at translation time, it is desirable to perform as many composition operations in Equation 1 as possible, ahead of time. The ideal situation is to compute $\mathcal{H}$ offline. At translation time, one need only compute the best path of $\mathcal{S} = \mathcal{I} \circ \mathcal{H}$. However, it can be very difficult to construct $\mathcal{H}$ given practical memory constraints. While this has been done in the past for word-level and constrained phrase-level systems [8], this has yet to be done for unconstrained phrase-based systems. In [7], this issue is tackled as the following.

First, we note that the source language segmentation transducer $\mathcal{P}$ explores all "acceptable" phrase sequences for any given source sentence. It is crucial that this transducer to be deterministic because this can radically affect translation speed and memory usage. In [7], we introduce an auxiliary symbol, denoted $\mathcal{EOP}$, marking the end of each distinct source phrase. By adding these artificial phrase boundary markers, each input sequence corresponds to a single segmented output sequence and the transducer becomes determinizable.

Secondly, while it may not be feasible to compute $\mathcal{H}$ in its entirety as a single FSM, we separate $\mathcal{H}$ into two pieces: the language model $\mathcal{L}$ and the translation model $\mathcal{M}$:

$$\mathcal{M} = \mathcal{M}in(\mathcal{M}in(\mathcal{D}et(\mathcal{P}) \circ \mathcal{T}) \circ \mathcal{W}) \qquad (2)$$

where $\mathcal{D}et$ and $\mathcal{M}in$ denotes the determinization and minimization operation respectively. In spite of the fact that $\mathcal{T}$ and $\mathcal{W}$ in (2) are not deterministic, and that minimization is formally defined on deterministic machines [9], in practice, we often find that minimization can help reduce the number of states of non-deterministic machines. It should also be noted that due to the determinizability of $\mathcal{P}$, $\mathcal{M}$ (the SIPL) in the above equation can be computed offline using a moderate amount of memory.

3.3.2. Search

In this approach, translation has been defined as finding the best path in $\mathcal{I} \circ \mathcal{M} \circ \mathcal{L}$. To address the problem of efficient computation, Zhou et al. [7] have developed a multilayer search algorithm. Specifically, we have one layer for each of the input FSM's: $\mathcal{I}$, $\mathcal{L}$, and $\mathcal{M}$. At each layer, the search process is performed via a state traversal procedure starting from the start state, and consuming an input word in each step in a left-to-right manner. This can be viewed as an optimized version of on-the-fly or dynamic composition integrated with a Viterbi search procedure. However, this specialized decoding algorithm has the advantage of not only significant memory efficiency and being possibly many times faster than general composition implementations found in FSM toolkits, but it can also incorporate information sources that cannot be easily or compactly represented using WFST's. For example, the decoder can allow us to apply the translation length penalties and phrase penalties to score the partial translation candidates during search.

We represent each state $\hat{S}$ in the search space using the following 7-tuple: (S_I, S_M, S_L, C_M, C_L, $\hat{h}$, $\hat{S}_{prev}$), where S_I, S_M, *and* S_L record the current state in each input FSM; C_M and C_L record the accumulated cost in $\mathcal{M}$ and $\mathcal{L}$ in the best path up to this point; $\hat{h}$ records the target word sequence labeling the best path up to this point; and $\hat{S}_{prev}$ records the best previous state. The initial search state $\hat{S}_0$ corresponds to being located at the start state of each input FSM with no accumulated costs. To reduce the search space, two active search states are merged whenever they have identical S_I, S_M and S_L values; the remaining state components are inherited from the state with lower cost. In addition, two pruning methods, histogram pruning and threshold or beam pruning, are used to achieve the desired balance

between translation accuracy and speed. The search algorithm is implemented using fixed-point arithmetic [7] for deployment on PDA devices that lack a floating point processor.

3.3.3. Translation experiment

We evaluated the proposed SIPL translation framework on two speech translation tasks from the government-sponsored TransTac program. This is bi-directional translation between English and a dialect of colloquial Arabic (DCA), and corpora statistics are shown in Table 2. The development sets were used to adjust the model and search parameters (e.g., pruning thresholds etc.). The translation experiments were done on the PDA with an average speed of 500 words per second, using less than 20 MB memory. Table 3 provides some sample DCA-English translation produced by our PDA system, and Table 4 shows the BLEU on the test set for a reference. More evaluation results will be discussed in [7]. From the performance observed in Table 3 and 4, we can see that the system achieved fast translation speeds yet retained high translation accuracy on the PDA platform.

Table 2. *English-DCA corpora statistics.*

	English	DCA
Training Set	366K sent. 7.9word/sent	366K sent. 5.5word/sent.
Vocabulary	24303 words	79960 words
Dev set	395 sent. 10.4 word/sent	200 sent. 6.5word/sent.
Test set	1856 sent. 9.1word/sent.	1856 sent. 6.3word/sent

Table 3. *Sample DCA-English Translation Results*

i just came here to visit my family
sure i understand that that's the way that used to be
how long do you have my identification
we had electricity yes in temporary we had electricity
about twenty meters further away from our house two stories built with bricks

Table 4. *Translation performance: Blue score.*

English->DCA	0.3883
DCA->English	0.5010

3.4. Text-to-Speech Synthesizer

Once an utterance is translated by the translation model, it is sent to the screen for display and to a text-to-speech (TTS) engine. IBM's embedded ViaVoice TTS system is used. Embedded TTS supports formant and concatenative TTS. Considering the limited resources available in a mobile device, formant TTS could be a reasonable choice. But high quality synthesized speech is vital for speech to speech communication. Therefore, compact male concatenative voices are used for both English and colloquial Arabic. The TTS system requires about 2 MB, additional 5 MB for English and 7 MB for Arabic voice fonts.

3.5. System Integration

As mentioned above, lots of effort has been made so that the LVCSR and translation modules fit within the relatively low computational resources available on the PDA. The major challenge of system integration is how to get all the components working together as a complete speech-to-speech translation system, without causing significant user delay.

To fully utilize the relatively large RAM on the PDA, we put the ASR engine along with all acoustic and language models in main memory. The TTS voice fonts are also loaded in main memory. Translation models are more than 400MB and are stored in the SD card. Memory mapping is used to map the models into a Windows CE large memory area during system initialization. Several techniques are developed to split components into several different processes, since in Windows CE, each process only has a maximum of 32 MB virtual address space.

Our whole system is packaged and stored in an SD card. An installation program was also developed so that when the user presses the reset button, the MASTOR system is installed automatically.

4. Summary

In this paper, we present our recent effect to develop a bi-directional speech-to-speech translation system on a PDA using Window CE or the popular Pocket PC platform. The system includes an HMM-based large vocabulary continuous speech recognizer using a tri-gram language model. The translation module maintains comparable accuracy and speed found in its desktop countpart. A newly introduced adaptation method allows the system to adapt to a new speaker and makes our system more robust in noisy environments.

5. Acknowledgements

The authors thank Yoshinori Tahara, Ruhi Sarikaya, Mohamed Afify, Hong-Kwang Kuo, Liang Gu, Wei Zhang, Etienne Marcheret, Tim Mathes and Max Tahir for their help and contribution to this work. This work is funded by the DARPA Transtac project.

6. References

[1] Zhou, B, Dechelotte D., and Gao, Y., "Two-way Speech-to-Speech Translation on Handheld Devices", in Proc. ICSLP2004.

[2] Gao, Y. et al., "MARS: A statistical Semantic Parsing and Generation-Based Multilingual Automatic Translation System", Machine Translation, 17: 185-212, 2002

[3] Povey, D. and Woodland, P.C., "Minimum Phone Error and I-Smoothing for Improved Discriminative Training", ICASSP'02.

[4] Afify, M. et al., "On the Use of Morphological Analysis for Dialectal Arabic Speech Recognition", in Proc. Interspeech 2006.

[5] Li, Y. et al, "Incremental on-line feature space mllr adaptation for telephony speech recognition", in Proc. ICSLP 2002.

[6] Balakrishnan, S. V., "Fast incremental adaptation using maximum likelihood regression and stochastic gradient descent", in Proc. EUROSPEECH 2003.

[7] Zhou, B., Chen, S. and Gao, Y., "A Memory Efficient Phrase-based Machine Translation Using Statistical Integrated Phrase Lattices", to be published.

[8] Zhou, B., Chen, S. and Gao, Y., "Constrained phrase-based translation using weighted finite-state transducers", in Proc. ICASSP 2005.

[9] Mohri, M., Pereira, F., and Riley, M., "Weighted finite-state transducers in speech recognition", Computer Speech and Language, 16(1):69--88. 2002