Chinese-English Mixlingual Automatic Speech Recognition System

Emily Hua
Dept. of Computer Science
Columbia University
Email: yh2901@columbia.edu

Kelly Chen
Dept. of Electrical Engineering
Columbia University
Email: kc3031@columbia.edu

Wendy Wang
QMSS
Columbia University
Email: ww2440@columbia.edu

***Abstract*—This paper presents a Mixlingual Automatic Speech Recognition System (ASR) for conversational Mandarin-English code-switching speech. We constructed the baseline deep neural network-hidden Markov model (DNN-HMM) on the SEAME corpus [1] and two free dictionary (THCHS30 and the CMU dictionary). We further improved the system by applying linear discriminant analysis (LDA), speaker adaptive training (SAT) and perturbation to get stronger GMM alignment and we experimented different types of neural networks including CNN and LSTM to further improve the performance. Our best system achieves a Mixed Error Rate (MER) of 39.6% on the SEAME test set, which is 9.6% relatively better than the baseline DNN hybrid system, and 18.4% better than the GMM-HMM system.**

Keywords*—*ASR, codeswitch, hybrid system.

I. INTRODUCTION

The ongoing globalization nourishes multilingual societies. The adaptation of English into Asian region's language usage is a prominent impact of its force. Multilingual speaking habits, for example Singaporean having Mandarin and English as their spoken languages, are associated with the code-switching phenomenon. Code-switching is defined as the "the alternate use of two or more languages in the same utterance or conversation" [2]. We want to tackle this code-switching challenge using hybrid models (ANN-HMM). Traditional ASR uses GMM-HMM system, which assumes Gaussian distribution of speech signals associated with each of HMM states. However, this fixed distribution assumption is not necessarily true. Artificial Neural networks come in to provide posterior for decoding without assuming any particular structure of the data. Our focus is on feature engineering to construct stronger GMM alignment and incorporate neural networks to improve system performance. That being said, we used a simple trigram language model based on SEAME transcript and two free dictionaries. The relatively weak language model produces a 19.5% out of vocabulary (OOV) words on the test set and 58.0% MER with the GMM-HMM system, which we later reduce the MER to 39.6%.

A. SEAME Corpus

Our data source is Mandarin-English Code-Switching in South-East Asia (SEAME) [1] acquired from Linguistic Data Consortium (LDC). It is a conversational code-switching corpus recorded from Singaporean and Malaysian speakers. There are 159 speakers and 192 hours of audio. After transcript-cleaning, which we will discuss in details in section II, around 50 hours of audio is kept to build our systems. The ratio of Chinese, English, Silence and Others is 44%, 26%, 21% and 7% in the original SEAME corpus [3]. After filtering out words with no phonetic pronunciation in our directories, the ratio between Chinese and English words in our lexicon is 29% vs.71%. We divide the corpus into training and testing set with test set consists of 14 speakers with balanced gender. Table 1 lists the statistics of SEAME Corpus in these two sets. As performance measure, we use Mixed Error Rate (MER), which applies to word error rate for English and character error rate for Mandarin.

Table 1. *Statistics of SEAME Corpus*

	Train set	Test set	Total
# Speakers	145	14	159
Duration (hours)	49.8	4.6	54.4
# Utterances	43221	3392	46613
# English Words	211925	14368	226293
# Chinese Words	332775	33404	366086
English OOV	9.7%	14.6%	10.0%
Chinese OOV*	4.4%	4.9%	4.5%

*Chinese OOV words were split into characters (see Data Preparation section), which led to a lower OOV rate.

B. Kaldi and Keras

The main tool that we used to build our systems is Kaldi [4], which is one of the best open sourced toolkit for speech recognition. It is implemented in C++ and has shell script as its interface. Although Kaldi has its own scripts to train neural networks, we also used Keras [5] for its easier interface to tune parameters.

II. DATA PREPARATION AND LANGUAGE MODEL

A. Lexicon Dictionaries

CMU dictionary (130K) was used for English words and THCHS30 dictionary (8.8K; an open sourced Chinese dictionary provided along with THCHS30 dataset) was used for Chinese words. To reduce OOV rate, we inserted new English words into the dictionary that can be split into known words (e.g., the phones of word handphone is attained by concatenating the phones of word hand and the phones of word phone).

The phone list was formed by collecting both English phones and Chinese phones. Similar phones of different language were not merged.

B. transcription Cleanup

In order to improve the quality of the transcription, we preprocessed the raw transcript on the following aspects:

- Basic cleaning, including (a) transform fullwidth forms to halfwidth forms, (b) split Chinese characters and English word if they are concatenated together, (c) remove all annotation signs that are not explained in documentation (% and ", e.g, %chelsia% → chelsia), (d) remove annotation sign for words from foreign language in case the word exists in English Dictionary (e.g., #sushi# → sushi), (g) fix wrong annotation (e.g., [ppl] → (ppl), ppl → (ppl)), (e) replace annotation with SIL index (e.g., [oh] → SIL2), (f) remove punctuation (e.g., ?).
- Chinese OOV split into characters, where OOV words are the words not contained in THCHS30 dictionary.
- SIL merged, where all the discourse particle (annotated by a pair of square brackets) are replaced with one silence symbol (SIL1) and all the hesitation and filled pause (annotated by a pair of round brackets) are replaced with the other silence symbol (SIL2).
- Spelling correction, including (a) correct misspelled phrases (e.g., abit → a bit) and (b) correct misspelled English words (e.g., avalable → available).
- Chinese OOV in test set split into characters, where OOV words are the words not contained in training dictionary (the dictionary created from words in the training set).

As some transcription contains poorly segmented sentences, only sentences with no more than 5 concatenated Chinese words are viewed as eligible for training and testing, which reduces the number of utterances to 43221.

Table 2. *Effect of Transcription Cleanup on WER under LDA-MLLT Acoustic Model*

Modification	WER
Chinese OOV split into characters	65.9%
Chinese OOV split into characters + SIL merged	64.3%
Chinese OOV split into characters + SIL merged + Spelling correction & Lexicon inserted	63.8%
Chinese OOV split into characters + SIL merged + Spelling correction & Lexicon inserted + Chinese OOV in test set split into characters	63.7%

*Word Error Rate (WER) based on English Words and Chinese words (not character)

C. Language Model

A simple trigram mixlingual language model is built from the SEAME training transcriptions. The perplexity of the language model is 220.93 on the test set.

Our training transcriptions might cause the language model to suffer from data sparsity, specially at the code-switching point. Some studies suggest interloping the existing language model with both English and Chinese monolingual language model to mitigate the problem, where those monolingual language models are trained on a much larger monolingual corpus [3]. During the current study, we refrained from using language model interpolation, because combining information from English-only corpus and Chinese-only corpus without the guide of prior knowledge is likely to destroy the specific rules that only exist in the mixlingual corpus. However, these approaches are worth trying as they may further improve the performance.

III. Feature Extraction

The first step in any automatic speech recognition system is to extract features of the audio signal. Feature extraction is used to reduce the dimensions and thus using less features to represent the speech characteristics. In Kaldi, after data preparation which has been discussed in the previous section, we need to make features.

A. MFCC Feature

Mel Frequency Cepstral Coefficients (MFCCs) is a feature most widely used in speech recognition to characterize the identity of speakers. It aims to capture some acoustic information from the audio signal.

To calculate MFCC, firstly a signal preprocessing is applied on a speech signal. It consists on a pre-emphasis filter to equalize the accurate size. A hamming window is applied to decrease the edge effects due to the windows cutting on each block. A hamming window has the following form:

$$w_H(n) = 0.54 - 0.46\cos(\frac{2\pi n}{n-1})n = 0, 1, ..., N-1 \quad (1)$$

A Fast Fourier Transform (FFT) is applied on the treated signal and smoothed by a series of triangular filters distributed on a Mel scale.Incorporating the Mel scale makes our features match more closely what humans hear. We can convert between Hertz (f) and Mel (m) using the following equations:

$$M(f) = 2595log_{10}(1 + f/700) \quad (2)$$

The log energies in the warped spectral coefficients are then processed via a discrete cosine transform (DCT) into a low dimensionality feature vector. It is important to note that DCT is a linear transformation and therefore it is not desirable to speech signals which are highly non-linear. [6] We will talk about another features which discard DCT process in the next section. The Kaldi documentation gives a more detailed description of how to calculate MFCCs. [7]

By using the Kaldi `make_mfcc.sh` recipe, we generate .ark (archive) files and .scp (script) files. The .scp files are in the form of:

```
<utterance-id> <extended-filename-of-features>
```

The extended-filename-of-features in the .scp file looks like `/.../mfcc/raw_mfcc_train.1.ark:24` and it means that open the archive file `/.../mfcc/raw_mfcc_train.1.ark`, fseek() to position 24, and read the data that's there. Each of the feature files contains a matrix and the dimension of the matrix is 13 by default (the length of the file in 10ms intervals). [8]

B. MFSC Feature

The main component of MFCC which is responsible for noise robustness is the filter bank. The filters smooth the spectrum, reducing variation due to additive noise across the bandwidth of each filter. [9] Filterbank features have advantages such as being less correlated and equally compressible.

For CNN, we have to use filterbank features but without the DCT performed, which are so-called MFSC (Mel-frequency spectral coefficients) features. The log-energy is computed directly from the mel-frequency spectral coefficients because the DCT projects the spectral energies into a new basis that may not maintain locality. Thus, for CNN training, we combine filter bank and pitch features together by utilizing the Kaldi recipe `make_fbank.sh` and `make_fbank_pitch.sh`.

In general, it is suggested that MFSC is better for neural network based models like DNN/CNN which helps to reduce WER, while MFCC is very popular with GMMs-HMMs. We compare the result of our experiment of using different features in the following section.

C. Delta+delta-deltas

HMM assumes conditional independence, where each hidden state depends only on the previous hidden state, and each observation depends only on the current hidden state. Since speech sequences are correlated in time, and it is infeasible to model the entire sequence, HMM is a practical way to model the temporal correlation. In order to loosen the conditional independence assumption of HMM, first and second differential parameters, the so-called delta and deltadelta parameters, are added to the static feature parameters. [10]

In Kaldi, we use `tran_deltas.sh` to train triphone models on top of `MFCC+delta+delta-deltas` features. The dimensionality after `add-deltas` is increased from 13 to 39.

D. LDA Feature

Linear Discriminant Analysis (LDA) is used as a dimensionality reduction method to reduce input dimensions from 117 (13 MFCC dimensions $\times$ (4+1+4) splice of frames) to 40 dimensions. It is a supervised generative model that maximizes the ratio of the between class variance to the average within class variance for each dimension. This objective yields an orthonormal transform such that the average within-class covariance matrix is diagonalised, which improves the diagonal covariance matrix assumption used in Gaussian Mixture Model (GMM) state-output distribution. [11]

During the training, we estimate a transform known as Maximum Likelihood Linear Transform (MLLT). MLLT allows sharing a few full covariance matrices across many distributions without storing and computing all, just so we don't have exploding parameters. This adds to basic MFCC features+GMM in a way that GMM systems use diagonal covariances matrices to get the emission probability. Given only diagonal covariance matrices are used, not full covariance, GMM assumes each element of the feature vectors (MFCCs) are independent. MLLT loosens this assumption by adding a few full covariances matrices to the system.

E. SAT Feature

When training a speaker independent system on a dataset generated by a large amount of speakers, acoustic models "have to 'waste' a large number of parameters encoding the variability between speakers rather than the variability between spoken words" [12]. Speaker Adaptive Training (SAT), a well studied method, is used to tackle this problem by projecting training data into a speaker-normalized space. Parameters of the acoustic models are then estimated in this new feature space. Trained in this way, acoustic model becomes speaker independent, and generalizes better to unseen speakers. [13]

In Kaldi, we use `train_sat.sh` script to build SAT on top of LDA+MLLT features. During decoding on the test dataset, `decode_fmllr.sh` is used to estimate and use feature space MLLR (fMLLR)-transforms based on alignments from a previous un-adapted decoding.

IV. Baseline Hybrid System Setup

A. DNN systems

In the baseline hybrid model, instead of using GMM, we use deep neural networks (DNN) to estimate emission probabilities for HMM. The output of the neural network is the probability of a phone class given the feature $P(s_j|x)$. In order to compute the emission probability $P(x|s_j)$ for HMM, we uses the Bayes Rule:

$$P(x|s_j) = P(s_j|x) \times P(x)/P(s_j) \tag{3}$$

which can be simplified as

$$P(x|s_j) \sim P(s_j|x)/P(s_j) \tag{4}$$

(this is okay, as $P(x)$ does not depend on the class s_j). That being said, we scale the neural net output by class priors $P(s_j)$ to get the emission probability for HMM.

DNN training will be greatly affected by the quality of the previously trained GMM-HMM model. A weak GMM-HMM will produce poor alignments, and results in a poor DNN at the end of the day. The standard Kaldi scripts for DNN training assume that we have trained a GMM-HMM and generated alignments for the training audio.

So, we prepare our input to the DNN by first train a flat start monophone model and then train a set of triphone models. Our baseline is to use MFCC-based features as the input. After training the triphone system using MFCC + Monophone + Deltas + LDA + MLLT (with 1000 leaves and 11 Gaussians per leaf), we feed the alignments from GMM-HMM model into DNN to get the emission probability.

Our baseline hybrid model is constructed using a DNN with 4 hidden layers of 1024 hidden units and ReLU activation function. This is trained using Pavan Kumar's Keras-Kaldi DNN script [5]. It takes around 4 hours to converge on Google Cloud's Tesla X80 GPU. While its corresponding GMM-HMM system has 58.0% MER, the baseline hybrid model achieves 49.2% MER.

V. Improved System Setup

A. DNN Systems with P-norm

As opposed to ReLU activation, we use *p-norm* (a variant of Maxout) activation function for our improved DNN. *p-norm* can be described as:

$$y = \|x\|_p = \left(\sum_i |x_i|^p \right)^{\frac{1}{p}} \tag{5}$$

where the vector x represents a small group of input.

A "normalization layer" that scales down "the whole set of activation is used to prevent the standard deviation from exceeding 1." [14]
σ is the uncentered standard deviation of x_i:

$$\sigma = \sqrt{1/K \sum_i (x_i)^2}, \tag{6}$$

where x_i is the input.
The nonlinearity is:

$$y_i = \begin{cases} x_i, & \text{if } \sigma \leq 1 \\ x_i/\sigma, & \text{if } \sigma > 1 \end{cases}$$

This is applied directly after *p-norm* (without a layer of weights in-between) to stabilize unbounded-output nonlinearities. This set up with p equals 2 out performs ReLU (with the same parameter settings). We used Kaldi's WSJ online recipe to train this system, and it achieves 47.7% MER.

B. DNN Systems with Perturbation

Perturbation is a technique used for data augmentation, It produces a warped time signal. Given an audio signal $x(t)$, time warping by a factor α gives the signal $x(\alpha t)$. From the Fourier transform of $x(\alpha t)$, $\alpha^{-1}\hat{x}(\alpha^{-1}w)$, the warping factor produces shifts in the frequency components by some proportional to the frequency [15]. In order to implement speed perturbation, we use the Kaldi `run_nnet2_perturb_speed.sh` script for this task. We first train a MFCC+ Monophone + Deltas + LDA + MLLT + SAT system for further usage. To modify the speed of a signal, we make two additional copies of the original training data and modify the speed to 90% and 110% of the original rate. MFCCs are used as input to the neural network. Due to the change in the length of the signal, the alignments for the speed perturbed data are regenerated using the GMM-HMM triphones model. Finally, we use p-norm with 4 hidden layers for DNN, which achieved 41.3% MER.

C. LSTM Systems

Long Short Term Memory networks (LSTM) is a type of Recurrent Neural Networks (RNN), that is capable of remembering long term dependencies. For building speech recognition system, LSTM is tempting as speech sequences are correlated on the time scale. LSTM is believed to outperform feed-forward DNN in a way that it can remember more through dynamically changing contextual window than DNN's limited temporal modelling on a fixed window.

We use Pavan Kumar's Keras-Kaldi LSTM script [5] to train a LSTM with 4 hidden layers, 1024 nodes and tahn activation function. It takes more than 30 hours to converge on Google Cloud's Tesla X80 GPU; at the time of writing this report, the LSTM is still running and has already produced lower loss on both training and evaluation set than feed-forward DNN with the same settings.

D. CNN Systems

The CNN model is trained mainly based on Kaldi's nnet CNN recipe, using the LDA+MLLT acoustic model as input.

It contains a 2-layer CNN with 1024 hidden units in each layer (the dimensionality of the convolutional kernel in the first layer is set to 8, the dim of the convolutional kernel in the second layer is set to 4, and the number of features representing pitch defaults to 3). This CNN pre-training is followed by a stack of 4-layer Restricted Boltzmann Machine (RBM) of 1024 hidden units in each layer, the output of which gets realigned.

RBM pre-training is used as the preparation to cross-entrophy training, which is done with DNN; with 0 hidden layer, this DNN training section serves to add a softmax layer to the pre-trained network. After that, this network is re-trained by 6 iterations of DNN with sMBR (state-level minimum Bayes risk), a sequence discriminative criterion. With sMRB as the objective function, instead of optimizing cross-entrophy, the network is trained to minimize the expected error corresponding to different granularity of state labels [16].

This entire system takes over 5 days to train on a GPU, and achieves 39.6% MER. Since the CNN system takes too much time to train, we focus more on the improvement of DNN models.

Table 3. *Model Comparison (MER) trained on Moderately Cleaned Transcript*

	MFCC	MFCC + pitch	Fbank
GMM-HMM (LDA)	58.0%	57.2%	
GMM-HMM (SAT)	53.1%		58.0%
Hybrid DNN (ReLU)	49.2%		
Hybrid DNN (p-norm)	47.7%	45.7%	
Hybrid DNN (p-norm) with Perturbation	41.3%		
Hybrid CNN (pre-softmax)	-	-	39.6%

*The modification of transcript includes splitting Chinese OOV into characters and merging SIL to 2 kinds.

** Pre-softmax activations are equivalent to log-posterior + C_frame.

E. SGMM

Subspace Gaussian Mixture Models (SGMM) might be another relatively conventional way to improve MER. In SGMM models, the number of model parameters can be reduced since they are derived from a low-dimensional model and speaker subspaces, given which only a small number of parameters would be able to derive the corresponding GMM models [17]. In practice, this approach was reported to contribute to a relative 3.3% improvement on English WER [18] and an absolute 6.25% improvement on Chinese WER [19].

We use Kaldi's `train_ubm.sh` and `train_sgmm2.sh` to build the SGMM system on top of the result of SAT + SAT, with 5000 leaves and 11000 Gaussians. It yields a fairly nice result within several hour's training and decoding, as low as 40.6%, a little higher than the result yielded by the CNN system which takes more than 5 days to train.

VI. Performance Results

A. Comparison

The MER is computed as follows:

$$MER = \frac{S + D + I}{N} \tag{7}$$

where N is the total number of words in the ground-truth transcription, and S, I, D denote the number of substitutions, insertions and deletion errors, respectively.

Table 4. *Model Comparison (MER) trained on Deeply Cleaned Transcript*

	MFCC	MFCC + pitch
GMM-HMM (SAT+SAT)	running	51.7%
GMM-HMM (SGMM)	running	40.6%
Hybrid DNN (p-norm) with Perturbation	44.5%	running

*The modification of transcript includes splitting Chinese OOV into characters, merging SIL to 2 kinds, spelling correction and lexicon insertion, and splitting Chinese OOV in test set into characters.

B. Error Breakdown

In order to gain more insight into the code-switching data, we look into detailed error rate on Mandarin and English languages, which is listed in Table 3. The percentage of each error is calculated as S/N, I/N and D/N.

Table 5. *Best Model (CNN) Errors %*

Error Type	English	Chinese	Total
Ins	2.2	2.0	4.1
Sub	9.6	18.3	27.8
Del	1.8	5.8	7.7

Firstly, we can see that substitution error, as expected, takes up the largest proportion. Secondly, Chinese substitution errors are more significant than English. We believe this is caused by the fact that there is 2 times more English words in the lexicon than Chinese words.

VII. Conclusion and Future Works

A. Conclusion

In this paper, we present our steps towards a large vocabulary continuous speech recognition system (LVCSR) for conversational code-switching speech. We experimented various network types to gain performance improvement on the hybrid ANN+HMM system based on a small language model with around 20% OOV. Our best CNN system achieves the lowest MER at 39.6% on the SEAME test set.

B. Future Works

1. Reduce OOV rate will for sure increase the performance. We found a dictionary that covers 112K Chinese words in our transcript, but it uses *pin yin* as its pronunciation. That results in over 1200 phones (previously 274 non-silenced phones) in our system, which blows up the 60G RAM when constructing the graphs. Vertically or horizontally scaling our machine, could resolve this issue.

2. Generating more transcription to construct a stronger language model could be another way to improve the performance. Statistical Machine Translation (SMT)-based text generation is investigated by Ngoc Thang Vu [3], which shows promising improvement in MER on the same SEAME codesiwtching corpus.

Acknowledgment

The authors would like to thank Professor Xiaodong Cui for his persistent guidance, Professor Liangliang Cao for the generous GPU credit support and his Alexa App suggestion, Markus Nussbaum-Thom and Chad DeChant for their suggestions on data source.

References

[1] D. Lyu, T. Tan, E. Chng and H. Li, An Analysis of a Mandarin-English Code-switching Speech Corpus: SEAME, Interspeech, Japan, 2010.

[2] Grosjean F.: Life with Two Languages. An Introduction to Bilingualism. Harvard University Press. 1982.

[3] Ngoc Thang Vu et al., "A first speech recognition system for Mandarin-English code-switch conversational speech", ICASSP, 2012.

[4] http://Kaldi-asr.org/doc/about.html

[5] https://github.com/dspavankumar/keras-Kaldi

[6] Fayek, Haytham. "Speech Processing For Machine Learning: Filter Banks, Mel-Frequency Cepstral Coefficients (Mfccs) And What's In-Between". N.p., 2016. Web. 21 Apr. 2016.

[7] http://Kaldi-asr.org/doc/feat.html

[8] http://Kaldi-asr.org/doc/data_prep.html

[9] Singh, Nilu, R. A. Khan, and Raj Shree. "MFCC And Prosodic Feature Extraction Techniques: A Comparative Study". International Journal of Computer Applications 54.1 (2012): 9-13. Web.

[10] Joseph Keshet, Samy Bengio, "Automatic Speech and Speaker Recognition: Large Margin and Kernel Methods", p85, Wiley Publications, 2009.

[11] Mark Gales and Steve Young, ”The Application of Hidden Markov Models in Speech Recognition”, p221, Foundations and Trends in Signal Processing, Vol.1, 2008.

[12] Mark Gales and Steve Young, ”The Application of Hidden Markov Models in Speech Recognition”, p253, Foundations and Trends in Signal Processing, Vol.1, 2008.

[13] Jiajie Miao et al, ”Speaker Adaptive Training of Deep Neural Network Acoustic Models using I-vectors”, p1, IEEE/ACM Transactions on Audio, Speech, and Language Processing, 2015.

[14] Xiaohui Zhang, Daniel Povey et al, ”Improving Deep Neural Network Acoustic Models Using Generalized Maxout Networks”, ICASSP, 2014.

[15] Tom Ko, Vijayaditya Peddinti, Daniel Povey, Sanjeev Khudanpur , ”Audio Augmentation for Speech Recognition”, Interspeech, 2015.

[16] Karel Vesely et al,”Sequence-discriminative training of deep neural networks”, p2, INTERSPEECH, 2013

[17] Lu, L. ”Subspace Gaussian mixture models for automatic speech recognition”, 2013.

[18] Povey, Daniel, Luk Burget, Mohit Agarwal, Pinar Akyazi, Kai Feng, Arnab Ghoshal, Ondej Glembek et al. ”Subspace Gaussian mixture models for speech recognition.” In Acoustics Speech and Signal Processing (ICASSP), 2010 IEEE International Conference on, pp. 4330-4333. IEEE, 2010.

[19] https://github.com/kaldi-asr/kaldi/blob/master/egs/hkust/s5/RESULTS