INTERSPEECH 2017
August 20–24, 2017, Stockholm, Sweden

Improving Deliverable Speech-to-text Systems with Multilingual Knowledge Transfer

Jeff Ma[1], Francis Keith[1], Tim Ng[2], Man-hung Siu[1], Owen Kimball[1]

Raytheon BBN Technologies, Cambridge, MA, USA

[1]{jeff.ma,francis.keith,man-hung.siu,owen.kimball}@raytheon.com

[2]estim.ng@gmail.com

Abstract

This paper reports our recent progress on using multilingual data for improving speech-to-text (STT) systems that can be easily delivered. We continued the work BBN conducted on the use of multilingual data for improving Babel evaluation systems, but focused on training time-delay neural network (TDNN) based chain models. As done for the Babel evaluations, we used multilingual data in two ways: first, to train multilingual deep neural networks (DNN) for extracting bottle-neck (BN) features, and second, for initializing training on target languages.

Our results show that TDNN chain models trained on multilingual DNN bottleneck features yield significant gains over their counterparts trained on MFCC plus i-vector features. By initializing from models trained on multilingual data, TDNN chain models can achieve great improvements over random initializations of the network weights on target languages. Two other important findings are: 1) initialization with multilingual TDNN chain models produces larger gains on target languages that have less training data; 2) inclusion of target languages in multilingual training for either BN feature extraction or initialization have limited impact on performance measured on the target languages. Our results also reveal that for TDNN chain models, the combination of multilingual BN features and multilingual initialization achieves the best performance on all target languages.

Index Terms: speech recognition, bottleneck feature, multilingual training

1. Introduction

Using multilingual data for improving speech recognition performance on low-resource languages has been one of the popular research areas in recent years [1][2][3][4][5][6][7][8][9]. Multilingual data has been mainly used in two ways. One is to train deep neural networks (DNN) to extract bottleneck (BN) features – outputs of a hidden layer that has significantly fewer nodes than other hidden layers (typically less than 100). The other is to train a DNN to initialize training on target languages that have limited data. BBN explored both approaches in building STT evaluation systems for the IARPA Babel program [10]. For the Babel evaluations, BBN trained DNNs on a multilingual data corpus for both generating BN features and initializing training on low-resource test languages [11][12]. The multilingual corpus consisted of 27 languages from both Babel and non-Babel data releases. The BN features helped improve STT performance significantly. Initializations with the multilingual DNNs also yielded sizable gains.

Following the Babel evaluation efforts, we have continued to improve STT accuracy with multilingual data. However, in addition to improving STT accuracy, which was the primary goal of Babel evaluations, we also focus on transferring technologies to build deliverable systems for customer applications. As one of such efforts, BBN has developed the Sage automatic speech recognition (ASR) toolkit [13], which integrates technologies from multiple resources, including proprietary sources, such as BBN's Byblos tools [14][15], and open source tools, such as Kaldi [16] and CNTK [17]. For deliverable systems, accuracy, simplicity and decoding speed are all important considerations. To speed up decoding, we adopt two strategies, single-pass decoding and down-sampling of input features. Many high-performing STT systems, including BBN Babel systems, run two-pass decoding for getting better accuracy. The first pass, called "un-adapted pass" sometimes, runs with speaker independent (SI) models. Outputs from the un-adapted decoding are used to build speaker adapted models, which are used to run the second pass – "adapted decoding". To reduce system complexity and improve decoding speed, we focus on building single-pass STT decoders.

Down-sampling input features has recently become popular in training STT systems, which can significantly speed up both training and decoding – roughly proportional to the down-sampling factor [18][19][20]. One approach to this direction is the down-sampling mechanism embedded in the "chain model training" strategy [20]. In this work we focus on developing systems with the chain model training approach.

The major contributions of our work in this paper include uses of multilingual data to improve accuracy of TDNN-based chain models, which cover both training of chain models on multilingual BN features and porting of multilingual chain model to target languages. Both improve STT performance significantly. Our results also reveal two important findings. The first is that initialization with TDNN models trained on multilingual data produces larger gains on target languages that have less training data, and the second is that for either BN feature extraction or initialization, inclusion of a target language in the multilingual training has limited impact on the target language STT performance. This finding suggests that a single multilingual BN feature extractor and a single multilingual initialization model is enough for all possible target languages.

This paper is organized as follows. Section 2 briefly describes the procedure we use to train TDNN-HMM hybrid systems. Section 3 elaborates the training methods using multilingual data. Section 4 shows the performance of systems trained with various methods. Section 5 concludes this work.

http://dx.doi.org/10.21437/Interspeech.2017-1058

2. Training TDNN-HMM hybrid systems

DNN-HMM hybrid systems have become the most popular basis for building STT systems. We adopt the HMM-TDNN hybrid approach to build STT systems and train TDNNs with the "chain model" strategy [20]. Compared to the traditional cross-entropy DNN training, "Chain model" training has two distinct differences: 1) use of the lattice-free maximum mutual information (LF-MMI) criterion for training; and 2) sub-sampling of input feature frames by a factor of three, which significantly reduces decoding time – by roughly three-fold. To enable the sub-sampling, a 2-state HMM topology is designed to allow one-frame traversal of the HMMs.

Before training of the chain models, a 3-state Gaussian mixture HMM system is first trained to create alignments for the training data. Then, the 3-state alignments are converted into 2-state alignments. Finally, a state-tying tree is created based on the 2-state alignments for chain model training. Our chain model training follows a procedure similar to Kaldi's recipe. We train TDNNs with the LF-MMI criterion, followed by the state-level minimum Bayesian risk (sMBR)-based sequence discriminative training.

Table 1: *Amounts of training and test data of four target languages*

Language	Train(hours)	Test(hours)
Geor	50	12.4
Egyp	20	1.9
Turk	83	9.3
Cant	110	9.5

In these experiments we train on four target languages, Georgian, Egyptian Arabic, Cantonese and Turkish. For notational simplicity, in the rest of this paper we denote them as "Geor", "Egyp", "Cant" and "Turk", respectively. The amounts of their training and test data are listed in Table 1. For each language the training data is doubled by one copy of augmentation created with varying speeds and adding Babel noises, as described in [21]. In this paper we report STT system performance (in terms of WER – word error rate) measured on the test sets given in Table 1.

3. Training with multilingual data

BBN used multilingual data twice in the training of the Babel evaluation systems [11][12]. A brief summary of the Babel training procedure is as follows,

1. Train a DNN bottleneck feature (DNN-BNF) extractor with the multilingual data
2. For each of the languages, extract BN features, train a Gaussian mixture model, estimate fMLLR transforms, and transform the BN features
3. Train a multilingual DNN (ML-DNN) with the fMLLR-transformed features from all the languages
4. Port the ML-DNN to a target language by replacing the block softmax output layer with the target softmax, then further train the whole DNN on the target language data

As can be seen, the multilingual data was used in Step 1 and 3. In Step 1, one multilingual DNN was trained for extracting BN features, and in Step 3, another multilingual DNN was trained for initializing training on target languages. The two multilingual DNNs were configured with the same type of output layers – block softmax – but different types of hidden layers. They both were trained on the same multilingual data.

Since we focus here on building single-pass systems, we do not need fMLLR-transformed features. In all our training, we therefore skip Step 2 (speaker adaptive training) and run steps 3 and 4 on the original BN features. Another difference is that for the Babel evaluations [12], BBN trained the ML-DNNs as typical DNNs with sigmoid hidden layers, but in the work reported here, unless otherwise specified, we train TDNN chain models, as described earlier.

3.1. Training multilingual DNN-BNF extractors

The multilingual data corpus used in the Babel training consists of 27 languages – 23 Babel (Georgian excluded) and 4 non-Babel languages, totaling about 2,300 hours. In training this data was doubled by adding one artificial copy created by adding random Babel noises and varying speed perturbations.

The DNN-BNF extractor was trained with four *p-norm* hidden layers. The BN layer was configured with a *p-norm* that had 80 output nodes, producing 80-dimensional BN features [11]. We denote this extractor as "ML-27".

To test whether adding more languages to the training of the DNN-BNF extractor would give extra benefits, we added conversational telephone speech data from 13 more languages to the 27-language corpus and trained a new DNN-BNF extractor without any changes to the network configuration. These 13 languages are: Chinese-accented-English (58), Dari (23), Egyptian (20), French (87), Georgian (50), Hebrew (55), Iraqi (49), Japanese (16), Korean (27), Malay (45), Russian (50), UK-English (45) and Urdu (45). The numbers in parentheses are hours of data from each language. The total is 520 hours, which is about one quarter of the amount from the 27 languages. Since this new feature extractor was trained on 40 languages, we denote it as "ML-40".

Table 2: *Performance (WERs) of TDNNs trained with the ML-27 and ML-40 BNF extractors on two languages, Georgian and Egyptian*

BNF-extractor	Geor	Egyp
ML-27	42.5	40.0
ML-40	42.5	39.8

We compared the ML-27 and ML-40 BNF extractors on two languages, Georgian and Egyptian, which are not in ML-27 but in ML-40. Using these two extractors, we generated BN features on the two languages and trained TDNN models[1] from the respective BN features. WERs measured on their test sets are shown in Table 2. As shown, the ML-40 extractor does not produce significant gains over the ML-27 extractor. Thus, no benefits are obtained even if the target languages are added in the multilingual training. This could suggest that the 27 languages are broad enough to cover variations from new languages. To avoid re-running experiments that we had done with the ML-27 extractor before, we used the ML-27 extractor in subsequent model training.

When training the ML-40 extractor, we used the same language weighting method as used in the ML-27 training. As noted in [11], this weighting method is probably sub-optimal. The work in [22] shows that unbalanced data from languages

[1] These TDNNs were trained with the Nnet2 tools, not TDNN chain models

could potentially hurt multilingual BN training. The data selection explored in [9] can be treated as another way for weighting language data. Data balance may become more important when more languages are used, which deserves further investigation in future work.

3.2. Training multilingual TDNN chain models

Different from the Babel ML-DNN training, we selected 11 languages to train ML-DNNs in our experiments. These languages, along with their data amounts, are shown in Table 3. Among the 11 languages, French is not one of the 27 languages used in the ML-27 BNF extractor and not a Babel language either. Another change is that the English data used here consists of 380 hours – a subset of the 2,300-hour English conversational telephone speech (CTS) corpus [14] – and is more than the 250 hours used in the Babel multilingual training. The use of the French data and more English data increases the amount of non-Babel data in our ML-TDNN chain model training, reducing the weighting of Babel languages in the training. The total amount of this 11-language corpus is 1,560 hours.

Table 3: *Languages and their data amounts (hours) used in multilingual TDNN training of networks*

Language	Amount	Language	Amount
English	380	Swahili	50
Mandarin	250	Pashto	98
Spanish	245	Tagalog	90
French	85	Turkish	83
Cantonese	110	Vietnamese	90
Haitian	80		

Unlike the Babel ML-DNN training, this ML-TDNN chain model training used a joint output layer across languages. To do so, we created a joint state clustering tree for all 11 languages. The procedure for creating the tree is: 1) For each language, rename its phonemes, except the "silence" phoneme, by adding a language-specific suffix to distinguish from other languages. Pool all the new phonemes together to create a joint phoneme set, in which only the "silence" phoneme is shared among the languages. 2) For each language, change the phoneme names in its dictionary accordingly. Merge all dictionaries into a joint one. 3) Train a Gaussian GMM model with the joint phoneme set and the joint dictionary on all the language data. Generate alignments for all the data. 4) Create a joint tree based on the alignments.

The joint tree we created as described above ended up with 16.3K nodes (or state classes). Hence, on average each language had about 1.5K nodes. The average number of classes per language is about 5K in the Babel ML-DNN training, so there are significantly fewer classes generated for each language in the joint tree. The number of output layer weights was decreased, resulting in reduced training time.

After training such a ML-TDNN, we port it to a target language with the following widely used fine-tuning approach: First, replace the TDNN output layer with the target output layer and randomly initialize the weights. Then, run one epoch to train the output layer weights on the target language data. Finally, fine-tune all network weights with a small learning rate.

In [6] it is observed that the use of block softmax at the output layer not only makes the multilingual training easier but also performs better. We did not have time to try a block softmax output layer in the ML-TDNN chain model training of this work, but hope to in future work.

4. Experimental results

As mentioned before, we built systems on four target languages: Georgian, Cantonese, Egyptian and Turkish. In this section we report performance of various systems measured on the test sets of these languages.

4.1. Training with MFCC plus ivector features

The default features for training TDNN-based chain models are high-resolution MFCC features (40-dimensional) plus 100-dimensional ivector features (denoted as "MFC+IV"). Each of the hidden layers of the TDNN are added, randomly initialized, and trained on a small portion of the data. Then the next layer is added, and this process is repeated until we have our target network. We denote this initialization as "Random". Following the LF-MMI training, the network was further trained using the sMBR criterion.

We had trained 3-state HMM-TDNN hybrid systems with the Kaldi Nnet2 toolkit before, by using cross-entropy (CE) plus sMBR training. A comparison between these systems and the LF-MMI trained 2-state HMM-TDNN systems on the four target languages is included in Table 4. All systems were trained with the MFCC+IV features. It is interesting to observe that the LF-MMI TDNNs outperform the CE TDNNs on all the target languages. This indicates that the "LF-MMI+sMBR" training method produces better accuracy than the "CE+sMBR" training.

Table 4: *WER(%) comparison of 3-state TDNN trained with cross-entropy and 2-state TDNN trained with LF-MMI on MFCC+IV features*

System	Training	Geor	Egyp	Cant	Turk
3-state	CE+sMBR	46.9	49.0	42.0	41.9
2-state	LF-MMI+sMBR	44.8	47.9	41.5	41.3

Before training multilingual chain models, we had trained a TDNN chain model on the English Fisher and Switchboard corpus, which consists of 2,300 hours of CTS data [14]. The idea was to take a model trained on a language for which we have a significant amount of data to initialize chain model training on target languages that have less data. With this English TDNN model for initialization, we carried out chain model training on the four target languages, following the fine tuning procedure described in Section 3.2. The learning rate for the fine-tuning was one-tenth of the rate used in the English training. We denote this cross-lingual (English-to-target) initialization as "CL-eng". Following the procedure described in Section 3.2, we trained a multilingual TDNN chain model with the MFC+IV features. Then, using this ML-TDNN for initialization, we re-trained the chain models on the target languages. We denote this initialization as "ML-mfc". Performance of all these systems, trained with MFC+IV features, is listed in Table 5.

First, we see that the cross-lingual "CL-eng" initialization provides significant gains over the "Random" initialization on all the target languages. The gains are 0.7, 2.2, 3.3 and 9.3 points on "Cant", "Turk", "Geor" and "Egyp", respectively. Recall that, the training data amounts of the four languages decrease accordingly. Thus, the "CL-eng" initialization yields more gains on languages with less data. Second, we observe

some improvements (and a larger improvement in the case of Cantonese) using the multilingual "ML-mfc" initialization over using the cross-lingual "CL-eng" initialization. Similar to the cross-lingual, the multilingual initialization also produces larger gains on languages that have less training data. An additional observation is that two of the four target languages (Cantonese and Turkish) were included in the 11 language multilingual training. However, the "ML-mfc" initialization does not yield larger gains on these two languages than on the other two, Georgian and Egyptian. This implies that the inclusion of a target language in the multilingual model training has little impact on the final model performance with multilingual initialization.

Table 5: *Performance (WERs) of TDNN systems trained with "MFC+IV" and "ML-BN" features and with various initializations*

Feature	Initialization	Geor	Egyp	Cant	Turk
MFC+IV	Random	44.8	47.9	41.5	41.3
MFC+IV	CL-eng	41.5	38.3	40.8	39.1
MFC+IV	ML-mfc	41.0	38.1	39.9	38.7
ML-27	Random	42.1	39.7	39.3	39.6
ML-27	ML-bn7	40.9	37.6	38.8	38.6
ML-27	ML-bn9	40.7	37.4	38.7	38.6

4.2. Effect of multilingual BN features

The above experiments showed the effects of multilingual (or cross-lingual) initialization using MFC+IV features. Next, we examine the case of using the ML-27 BN features. First, we used the ML-27 BNF extractor to extract BN features on the four target language data and then trained a TDNN chain model with the "Random" initialization for each of the target languages. Next, we used the ML-27 extractor to extract BN features for the 11 languages, and trained two multilingual TDNNs with the BN features. One was trained with the default configuration (7 hidden layers) and the other with a larger configuration (9 hidden layers). All the hidden layers in both configurations had the same sizes (576 nodes). We trained the larger one to see the effect of using a larger multilingual model for initialization. Finally, we initialized training on the target languages with these two ML-TDNNs. We denote these two initializations as "ML-bn7" and "ML-bn9", respectively. Performance of these systems, trained on the ML-27 BN features, is given in Table 5.

First, comparing between the two types of features, we see that with the same "Random" initialization, the "ML-27" systems reduce WERs by more than 2 points over the "MFC+IV" systems. Comparing to the "MFC+IV" systems trained with the "ML-mfc" initialization, we see a mixed picture where the "Random"-initialized "ML-27" systems produce better performance on Cantonese, but worse on the other three languages.

Second, we can observe: there are no significant differences between the ML-bn7 and ML-bn9 initializations and they both yield further gains over the "Random" initialization on all the target languages. Similar to observed on MFC+IV features, the ML-TDNN initialization on the ML-27 BN features also produces larger gains on languages that have less training data. The multilingual initialization on the ML-27 BN features yields less gains than that on the "MFC+IV" features.

Finally, the training with the ML-27 BN features and the ML-bn9 initialization produce the best performance on all the target languages (though only slightly so on Turkish).

As previously mentioned, decoding speed is an important focus of the current work. We ran decoding with the multilingual initialized ML-27 BN systems on "x86_64" machines with 2.8GHz CPUs. The speeds measured on the target languages fall around *0.4-0.5* real-time (*xRT*). We observed that the BN feature computation takes about *0.1xRT*. This could be a factor to consider if a deployed recognizer is needed to run faster than *0.1xRT*. In such a situation the BNF-DNN can be optimized to have a smaller structure and faster processing.

The BBN Babel Georgian evaluation system achieved a 40.7% WER [12] before the web data was added, which is the closest system that we can compare with. The best Georgian system trained here yields the same WER. This suggests that with the improved training techniques, we simplified the system and sped up decoding without suffering on accuracy. As reported in [12], adding the web data greatly reduced out-of-vocabulary words and produced large gains (>3 points). In future work, we will verify if web data yields similar gains on the systems reported in this paper.

5. Conclusions

In this paper we investigated a number of aspects of multi-lingual DNN training. First, we found that adding more languages on top of the ML-27 BNF DNN training did not provide further gains. This implies that the multilingual BNF training could have already saturated with the 27 languages. Better data balance approaches may become more important when more languages are used, which deserves further investigation in future work. Besides data balance, other issues, such as input features and the DNN structure, may also warrant investigation.

Second, our results show that both the cross-lingual and multilingual initializations improve performance of the TDNN-based chain models greatly over their monolingual counterparts. Both produce larger gains on target languages that have less training data, but multilingual initialization seems to always yield better performance than cross lingual initialization. Additionally, in the case of multilingual initialization, whether the target language is included in the multilingual set has limited impact on the final accuracy of the target language system.

Third, the TDNN chain models trained on the multilingual BN features perform significantly better than their counterparts trained on the MFCC plus ivector features. The multilingual model initialization in this scenario also produces larger gains on languages that have less data. However, gains from the multilingual initialization are smaller than that obtained in the scenario using the MFCC+IV features. Since the multilingual BNF extractor was trained on a large number of speakers from various resources, such as different languages and releases, speaker variations carried in the BN features are most likely reduced. This alleviates the overfitting problem when training DNNs with these BN features.

Finally, as shown in our results, the combination of multilingual BN features and multilingual TDNN initialization – in particular TDNN chain models trained on multilingual BN features with multilingual initialization – yielded the best performance on all four target languages.

6. References

[1] A. Stolcke et al., "Cross-domain and cross-language portability of acoustic features estimated by multilayer perceptrons," in ICASSP 2006.

[2] S. Thomas, S. Ganapathy, and H. Hermansky, "Multilingual MLP features for low-resource LVCSR systems," in Proc. ICASSP, 2012.

[3] Z. Tuske, J. Pinto, D. Willett and R. Schluter, "Investigation on cross- and multilingual MLP features under matched and mismatched acoustical conditions," in Proc. ICASSP, 2013.

[4] K.M. Knill, M.J.F. Gales, et al., "Investigation of multilingual deep neural networks for spoken term detection," in Proc. of ASRU 2013.

[5] Jui-Ting Huang et al., "Cross-language knowledge transfer using multilingual deep neural network with shared hidden layers," in Proc. ICASSP, 2013.

[6] Q. B. Nguyen, J. Gehring, et al., "Multilingual shifting deep bottleneck features for low-resource ASR", in Proc. ICASSP 2014.

[7] Z. Tuske, R. Schluter, and H. Ney, "Multilingual MRASTA features for low-resource keywork search and speech recognition systems", in Proc. ICASSP 2014.

[8] F. Grezl, M. Karafiat and K. Vesely, "Adaptation of multilingual stacked bottle-neck neural network structure for new languages," in Proc. ICASSP 2014.

[9] E. Chunangsuwanich, Y. Zhang and J. Glass, "Multilingual data selection for training staked bottleneck features", in ICASSP 2016.

[10] Mary Harper, "IARPA Babel Program," http://www.iarpa.gov/Programs/ia/Babel/babel.html.

[11] T. Alumae, S. Tsakalidis and R. Schwartz, "Improved multilingual training of stacked neural network acoustic models for low resource languages," in INTERSPEECH 2016, September 8-12, 2016, pp3883-3887.

[12] T. Alumae, D. Karakos, W. Hartmann, R. Hsiao, etc., "The 2016 BBN Georgian telephone speeh keyword spotting system", to be appearing in ICASSP 2017.

[13] R. Hsiao, R. Meermeier, T. Ng, Z. Huang, M. Jordan, et al., "Sage: The new BBN speech processing platform," in INTERSPEECH 2016.

[14] R. Prasad, S. Matsoukkas, C.-L. Kao, J. Ma, etc, "The 2004 BBN/LIMSI 20xRT English conversational telephone speech recognition system", in INTERSPEECH 2005, pp 1645-1648.

[15] T. Ng, R. Hsiao, L. Zhang, D. karakos, etc., "Progress in the BBN keyword search system for the DARPA RATS program," in INTERSPEECH 2014, pp. 959-962.

[16] D. Povey, A. Ghoshal, G. Boulianne, L. Burget, O. Glembek, etc., "The Kaldi speech recognition toolkit," in ASRU 2011.

[17] D. Yu, A. Eversole, M. Seltzer, K. Yao, B. Guenter, et al., "An introduction to computational networks and the computational network toolkit,", Microsofet Research, tech. Rep., 2014.

[18] G., Pundak and T. N. Sainath, "Lower frame rate neural network acoustic models", INTERSPEECH 2016.

[19] V. Peddinti, D. Povey and S. Khudanpur, "A time delay neural network architecture for efficient modeling of long temporal contexts," in Proc. INTERSPEECH 2015.

[20] D. Povey, V. Peddinti, D. Galvez, P. Ghahrmani, et al., "Purely sequence-trained neural networks for ASR based on lattice-free MMI", in ICASSP 2016.

[21] W. Hartmann, T. Ng, R. Hsiao, S. Tsakalidis, and R. Schwartz, "Two-stage data augmentation for low-resourced speech recognition," in INTERSPEECH 2016.

[22] F. Grezl, E. Egorova and M. Karfiat, "Study of large data resources for multilingual training and system porting", in Proceedings of 5th Workshop on Spoken Language Technology for Under-resource languages, SLTU 2016, May 2016, Yogyakarta, Indonesia.