EXTRACTING DEEP BOTTLENECK FEATURES USING STACKED AUTO-ENCODERS

Jonas Gehring[1] *Yajie Miao*[2] *Florian Metze*[2] *Alex Waibel*[1,2]

[1] Interactive Systems Lab, Karlsruhe Institute of Technology; Germany
[2] Language Technologies Institute, Carnegie Mellon University; Pittsburgh, PA; USA
jonas.gehring@kit.edu ymiao@cs.cmu.edu

ABSTRACT

In this work, a novel training scheme for generating bottleneck features from deep neural networks is proposed. A stack of denoising auto-encoders is first trained in a layer-wise, unsupervised manner. Afterwards, the bottleneck layer and an additional layer are added and the whole network is fine-tuned to predict target phoneme states. We perform experiments on a Cantonese conversational telephone speech corpus and find that increasing the number of auto-encoders in the network produces more useful features, but requires pre-training, especially when little training data is available. Using more unlabeled data for pre-training only yields additional gains. Evaluations on larger datasets and on different system setups demonstrate the general applicability of our approach. In terms of word error rate, relative improvements of 9.2% (Cantonese, ML training), 9.3% (Tagalog, BMMI-SAT training), 12% (Tagalog, confusion network combinations with MFCCs), and 8.7% (Switchboard) are achieved.

Index Terms— Bottleneck features, Deep learning, Auto-encoders

1. INTRODUCTION

The two main approaches for incorporating artificial neural networks (ANNs) in acoustic modeling today are hybrid systems and tandem systems. In the former, a neural network is trained to estimate the emission probabilities for Hidden Markov Models (HMM) [1]. In contrast, tandem systems use neural networks to generate discriminative features as input values for the common combination of Gaussian Mixture Models (GMM) and HMMs. This is done by training a network to predict phonetic targets, and then either using the estimated target probabilities ("probabilistic features" [2]) or the activations of a narrow hidden layer ("bottleneck features", BNF [3]). Those features are usually fused with standard input features and decorrelated for modeling with GMMs.

In the field of machine learning, deep learning deals with the efficient training of deep neural networks (DNN), with algorithms generally inspired by the greedy, unsupervised, layer-wise pre-training scheme first proposed by Hinton et al. [4]. While Hinton et al. used a deep belief network (DBN) where each layer is modeled by a restricted Boltzmann machine (RBM), later works showed that other architectures like auto-encoders [5] or convolutional neural networks [6] are suitable for building deep networks using similar schemes as well.

Neural networks have been successfully used for acoustic modeling over two decades ago [7],[8], but had been mostly abandoned in favor of GMM/HMM acoustic models throughout the late 1990s. However, the ability to train networks with millions of parameters in a feasible amount of time has caused a renewed interest in connectionist models [9]. Recently, deep neural networks have been used with great success in hybrid DNN/HMM systems, resulting in strong improvements on challenging large-vocabulary tasks such as Switchboard [10] or Bing voice search data [11].

2. RELATED WORK

While bottleneck features have been used in speech recognition systems for some time now, only few works on applying deep learning techniques to this task have been published.

In 2011, Yu & Seltzer applied a deep belief network as proposed by Hinton et al. for extracting bottleneck features, with the bottleneck being a small RBM placed in the middle of the network [12]. The network was pre-trained on frames of MFCCs including deltas and delta-deltas, and then fine-tuned to predict either phoneme or senone targets. They found that pre-training the RBMs increases the accuracy of the recognition system, and that additional strong improvements can be achieved by using context-dependent targets for supervised training. However, they noted that possibly due to their symmetric placement of the bottleneck layer, increasing the number of layers in the network to more than 5 did not improve recognition performance any further. In a more recent work, it was also argued that RBMs are not suitable for modeling decorrelated data like MFCCs [13].

Sainath et al. introduced DBN training in a previously proposed architecture based on training an auto-encoder on phonetic class probabilities estimated by a neural network [14],[15]. In their work, they first trained a stack of RBMs for classification of speaker-adapted PLP features and applied a 2-step auto-encoder to reduce the output of the resulting DBN to 40 bottleneck features. These features out-performed a strong GMM/HMM system using the same input, but they found that performance gains are higher when training systems on little data.

This work proposes a different approach that profits from increasing the model capacity by adding more hidden layers, and enables the supervised training of the bottleneck layer in order to retrieve useful features for a GMM/HMM acoustic model. Instead of pre-training the layers with restricted Boltzmann machines, we use auto-encoders which are straightforward to setup and train.

3. MODEL DESCRIPTION

The architecture proposed for bottleneck feature extraction is illustrated in Figure 1. A deep neural network consisting of a stack of auto-encoders is first pre-trained on frames of speech data in a layer-wise, unsupervised manner. This process follows the standard

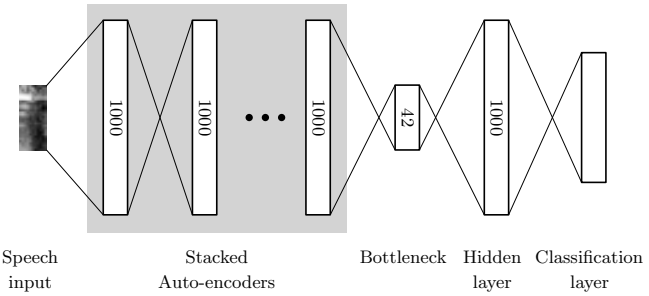

Fig. 1. Proposed architecture

scheme for pre-training a network that might be used for a classification task later. Afterwards, the bottleneck layer followed by a hidden and a classification layer are added to the network. The whole network is then fine-tuned in order to predict the phonetic targets attached to the input frames. Since there are potentially many hidden layers between the input data and the bottleneck layer, we call features extracted this way "deep bottleneck features" (DBNF).

For pre-training the stack of auto-encoders, we used denoising auto-encoders as proposed for learning deep networks by Vincent et al. [16]. This model works like a standard auto-encoder (or auto-associator) network, which is trained with the objective to learn a hidden representation that allows it to reconstruct its input. The difference is that in order to force even very large hidden layers to extract useful features, the network is forced to reconstruct the original input from a corrupted version, generated by adding random noise to the data. This corruption of the input data can be formalized as applying a stochastic process q_D to an input vector $\boldsymbol{x}$:

$$\tilde{\boldsymbol{x}} \sim q_D(\tilde{\boldsymbol{x}}|\boldsymbol{x})$$

The standard approach also used in this work is to apply masking noise to the data by setting a random fraction of the elements of $\boldsymbol{x}$ to zero. Using the weight matrix $\boldsymbol{W}$ of the hidden layer, the bias vector $\boldsymbol{b}$ of the hidden units and a non-linear activation function σ_y, the hidden representation $\boldsymbol{y}$, or the encoding, is then computed as follows:

$$\boldsymbol{y} = \sigma_y \left(\boldsymbol{W}\tilde{\boldsymbol{x}} + \boldsymbol{b}\right)$$

In a model using tied weights, the weight values are used for both encoding and decoding. The reconstruction $\boldsymbol{z}$ is thus obtained using the transposed weight matrix and the visible bias vector $\boldsymbol{c}$, again followed by a non-linear function σ_z:

$$\boldsymbol{z} = \sigma_z \left(\boldsymbol{W}^T\boldsymbol{y} + \boldsymbol{c}\right)$$

The resulting auto-encoder is then trained to reconstruct its original input $\boldsymbol{x}$ from the randomly corrupted version, which is done using standard back-propagation to minimize a corresponding error term, or loss function $L(\boldsymbol{x}, \boldsymbol{z})$. For modeling real-valued speech data with the first denoising auto-encoder, we use the mean squared error loss:

$$L_2\left(\boldsymbol{x}, \boldsymbol{z}\right) = \sum_i \left(x_i - z_i\right)^2$$

After the first auto-encoder has been trained, another one is trained to encode and reconstruct the hidden representation of the first one in a similar fashion. This time, the first auto-encoder computes its encoding from the uncorrupted version of the input vector, and corruption is applied to the input of the model being currently trained only. Since the second auto-encoder now models the probabilities of hidden units in the first auto-encoder being active, we train it using the cross-entropy loss as suggested in [16]:

$$L_H\left(\boldsymbol{x}, \boldsymbol{z}\right) = \sum_i x_i \log z_i + \left(1 - x_i\right) \log z_i$$

After training a stack of auto-encoders in this manner, a feed-forward neural network is constructed by connecting a small bottleneck layer to the top auto-encoder, followed by another hidden layer and the classification layer. All those layers are not being pre-trained, but instead initialized using random weights sampled uniformly out of a small range, which was also done to initialize the auto-encoder weights. The classification layer employs a softmax activation function for estimating class probabilities, and the whole network is trained using standard backpropagation.

4. EXPERIMENTAL SETUP

4.1. Baseline Systems

Baseline system setup and evaluation of trained networks were done using the JANUS recognition toolkit [17] and IBIS decoder [18] in a similar configuration as described in our previous works on bottleneck features [19]. For the baseline, samples consisting of 13 MFCCs were extracted and stacked with 11 adjacent samples, resulting in a total of 143 coefficients. LDA was applied to compute the final 42-dimensional feature vectors for the recognition system. Acoustic model training was performed differently for each system used for evaluation; please refer to section 4.4 for individual descriptions and resulting baseline performances.

4.2. Network Training

In our experiments, we extracted deep bottleneck features from both MFCCs and log mel scale filterbank coefficients (lMEL). The network input for MFCCs consisted of the same 143 coefficients as for the baseline system, sampled from the data with a context-independent model. When using lMEL features, 30 coefficients were extracted from the spectrogram and stacked in the same way, thus forming a 330-dimensional feature vector. During supervised fine-tuning, the neural network was trained to predict monophone states.

For pre-training the stack of auto-encoders in the proposed architecture, mini-batch gradient descent with a batch size of 64 and a learning rate of 0.01 was used. Input vectors were corrupted by applying masking noise to set a random 20% of their elements to zero. Each auto-encoder contained 1000 hidden units and received 4 million updates before its weights were fixed and the next one was trained on top of it. Limiting the training time by model updates rather than by epochs was done in order to be able to compare the influence of using different datasets for pre-training.

The remaining layers were then added to the network, with the bottleneck consisting of 42 units. Again, gradients were computed by averaging across a mini-batch of training examples; for fine-tuning, we used a larger batch size of 256. Supervised training was done for 50 epochs with a learning rate of 0.05. After each epoch, the current model was evaluated on a separate validation set, and the model performing best on this set was used in the speech recognition system afterwards. The training of auto-encoder layers and neural networks was done on GPUs using the Theano toolkit [20].

4.3. Bottleneck Features

The 42 output values of the bottleneck layer of 11 adjacent frames were stacked and reduced to a 42-dimensional feature vector using LDA. Using this feature vector, a context-dependent system was trained starting from a context-independent MFCC baseline system as described in section 4.1.

4.4. Corpora Description and Baseline Performance

Initial experiments for selecting optimal network layouts and hyperparameters were performed on a development system, which was trained on 44 hours of Cantonese conversational telephone speech. This corpus was recently released as the IARPA Babel Program [21] Cantonese language collection babel101-v0.1d. Other releases from this program used in this work are the Cantonese language collection babel101-v0.4c containing 80 hours of actual speech and the Tagalog language collection. The latter comes in two versions: babel106-v0.2f consisting of 69 h of conversational telephone speech, and the subset babel106b-v0.2g-sub-train with 9 h of speech.

All baseline systems used MFCC input data and were among initial builds on these recent and small corpora. Both Cantonese systems used GMM/HMM acoustic models that were ML-trained only. For Cantonese babel101-v0.1d, the baseline achieved a character error rate (CER)[1] of 71.5% by using a 3-gram language model extracted from transcriptions included in the babel101-v0.1d data. The baseline system for the full dataset babel101-v0.4c employed a 3-gram language model interpolating the main corpus and Cantonese Wikipedia articles, which resulted in 66.4% CER.

In contrast to the setups for Cantonese, the Tagalog babel106-v0.2f and babel106b-v0.2g-sub-train systems used Boosted Maximum Mutual Information (BMMI) discriminative training and speaker adaptive training (SAT) for its acoustic models. A similar language model as for Cantonese babel101-v0.4c was used, generated from the corpus transcriptions and text from Tagalog Wikipedia articles. This resulted in a WER of 64.3% for the full version trained on 69 hours and 72.1% WER for the babel106b-v0.2g-sub-train subset.

For evaluating the proposed setup on a larger task in a language more frequently used in speech recognition development, the Switchboard dataset was used [22]. Similar to the Cantonese systems, acoustic models were set up using ML training, this time on 300 hours of speech. For decoding, a 3-gram language model was generated from the transcriptions included in the dataset, which resulted in a WER of 39.0%.

5. RESULTS

In first experiments on Cantonese babel101-v0.1d, we compared different network architectures and inputs. For the input data, we found that extracting deep bottleneck features from lMEL instead of MFCC data resulted in consistently better recognition performance of about 2% CER absolute. This is consistent with the findings for deep belief networks in [13], and although we did not perform additional experiments to confirm that the larger dimensionality of the lMEL input vector is not the sole reason for the performance gap, we decided to use lMEL features for further experiments. Using more than 1000 hidden units for each of the auto-encoder layers did not result in improved recognition rates.

[1] Since Cantonese lacks a clear definition of a word, character error rate was used instead of the usual word error rate for evaluating the respective systems.

AE Layers	1	2	3	4	5
Network trained on 44 h					
With Pre-tr.	67.9	66.9	66.8	**66.0**	66.5
No Pre-tr.	69.2	**68.7**	70.8	72.0	71.1
Network trained on 80 h					
With Pre-tr.	67.2	64.7	63.8	**63.1**	63.2
No Pre-tr.	66.4	**65.9**	66.7	67.6	66.7
Pre-training on 80 h, fine-tuning on 44 h					
With Pre-tr.	68.7	66.4	66.5	**65.5**	66.1

Table 1. Performance of the Cantonese babel101-0.1d system (in % CER) on features generated from different networks. When trained on MFCCs, the system achieved 71.5% CER.

5.1. Importance of Pre-training

In order to determine whether pre-training of the stack of auto-encoder in front of the bottleneck was necessary, we evaluated the features generated by networks of different depth, on different amounts of data and with as well as without unsupervised pre-training.

Table 1 provides a listing of all those experiments. When training networks on the 44 hours of Cantonese data used in the development system, it can be seen that if pre-training is applied, recognition performance improves as more auto-encoder layers are added. If the neural network is trained starting from the usual random initialization, the character error rate increases significantly for deeper architectures.

The same experiment was repeated on the development system with networks trained on the full Cantonese babel101-v0.4c dataset. As before, pre-training turned out to be essential for the network to benefit from additional auto-encoder layers. However, the difference between DBNFs from pre-trained and purely supervised trained networks was smaller in terms of CER. This may indicate that unsupervised, layer-wise pre-training is particularly helpful if little data is available, and that this effect decreases with large amounts of data. Similarly, in their work regarding hybrid DNN/HMM models trained on the 300 h Switchboard benchmark, Seide et al. found that pre-training lowers the final error rate, but only by a small margin [10].

5.2. Usage of Unlabeled Data

Since unsupervised pre-training was found to be an essential part in achieving good recognition performance, we investigated whether the network could make use of additional, unlabeled data. Especially for conversational speech data, labeling is time-consuming and error-prone, and being able to increase recognition performance by using unlabeled data would be highly beneficial.

For Cantonese, we used the full 80 hours of data from babel101-v0.4c for pre-training the auto-encoders but fine-tuned the network on the babel101-v0.1d data as before. As shown in the last row of Table 1, this resulted in additional, small improvements of up to 0.75% relative over DBNFs from networks trained on the babel101-v0.1d subset only.

On Tagalog, we used the existing separation between the full babel106-v0.2f and the babel106b-v0.2g-sub-train version, so the ratio for unlabeled to labeled data was almost 5 times as high as for the experiment on Cantonese. As shown in the two last rows of Table 3, the word error rate could be lowered by 1.3% relative (for the BMMI-SAT system) by pre-training the network on the full 69 hours.

5.3. Evaluation on Larger Datasets

Further evaluation of the proposed architecture was done on the full datasets described in section 4.4. For the Cantonese babel101-v0.4c dataset consisting of 80 hours, the best result could be achieved with a network containing 4 pre-trained auto-encoder layers. As shown in Table 2, the character error rate could be reduced from 66.4% to 60.3%, which corresponds to 9.2% relative improvement.

System	MFCC	DBNF (by AE Layers)					
		1	2	3	4	5	6
CER (%)	66.4	63.6	62.0	60.9	60.3	60.5	61.1

Table 2. Character error rates of the Cantonese babel101-0.4c system with MFCC and DBNF input features.

The Tagalog system used a speaker-adapted acoustic model that was discriminatively trained, so we were interested in whether the performance gains on the ML systems would persist in a more advanced setting. On this task, networks containing 5 auto-encoder layers were found to extract the best features, so this architecture was used for subsequent experiments. The top of Table 3 shows that with standard ML training, an improvement of over 12% relative could be achieved when using DBNFs, lowering the WER from 70.1% to 61.5%. With BMMI/SAT, the relative gain between 64.3% WER on MFCCs and 58.3% WER on DBNFs was 9.3%, which is still high but lower than the one achieved on the ML system. Similar results were obtained on the babel106b-v0.2g-sub-train subset.

By performing a confusion-network combination of the MFCC and DBNF systems trained with BMMI/SAT, the recognition accuracy could be raised to 56.6% WER for the full system and 66.9% WER for the system trained on the subset.

AM Training	ML	BMMI-SAT	Combination
Full system (69 h)			
MFCC	70.1	64.3	56.6
DBNF	61.5	58.3	
Limited system (9 h)			
MFCC	77.4	72.1	66.9
DBNF-pre-full	70.4	68.9	
DBNF	71.7	69.8	n/a

Table 3. Results with deep bottleneck features from a 5-auto-encoder network on Tagalog babel106-v0.2f (full) and babel106b-v0.2g-sub-train (limited) systems (in % WER). The DBNF-pre-full network was pre-trained on the full dataset and fine-tuned on the limited dataset.

On Switchboard, we trained networks with 4 auto-encoders on subsets of the data and used the bottleneck features to set up a context-dependent system on the full 300 hour dataset. Using the features from a network trained on 60 hours of speech lowered the word error rate from 39.0% to 36.1% (7.4% relative). Doubling the amount of training data for the neural network resulted in further improvement and produced a WER of 35.6%, which is a 8.7% relative gain over the MFCC baseline.

6. CONCLUSION

In this work, we have proposed a new setup for extracting bottleneck features from deep neural networks and have shown its ability to achieve significant improvements over a number of MFCC baseline systems on different datasets. The model used is able to produce further gains by pre-training the stack of auto-encoders on more unlabeled data. This turned out to be more useful if only very little labeled data can be used for supervised fine-tuning and system training. We have also demonstrated that denoising auto-encoders are applicable for modeling speech data and initializing deep networks.

The results presented support hypotheses from earlier works in that log mel scale coefficients are more suitable input features for deep neural networks than MFCCs and that pre-training is generally beneficial but especially crucial when not much data is available.

Further work will deal with optimizing input feature vectors and system combinations as it is currently being done with standard bottleneck features. It might be interesting to compare denoising auto-encoders with the more widely used RBMs for pre-training and to fine-tune the network to predict context-dependent targets as was done in previous works.

7. ACKNOWLEDGMENTS

Supported in part by the Intelligence Advanced Research Projects Activity (IARPA) via Department of Defense U.S. Army Research Laboratory (DoD/ARL) contract number W911NF-12-C-0015. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation thereon. Disclaimer: The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of IARPA, DoD/ARL, or the U.S. Government.

8. REFERENCES

[1] H. Bourlard and N. Morgan, *Connectionist speech recognition: a hybrid approach*, vol. 247, Springer, 1994.

[2] H. Hermansky, D.P.W. Ellis, and S. Sharma, "Tandem connectionist feature extraction for conventional HMM systems," in *Acoustics, Speech, and Signal Processing, 2000. ICASSP'00. Proceedings. 2000 IEEE International Conference on*. IEEE, 2000, vol. 3, pp. 1635–1638.

[3] F. Grézl and P. Fousek, "Optimizing bottle-neck features for LVCSR," in *Acoustics, Speech and Signal Processing, 2008. ICASSP 2008. IEEE International Conference on*. IEEE, 2008, pp. 4729–4732.

[4] G.E. Hinton, S. Osindero, and Y.W. Teh, "A fast learning algorithm for deep belief nets," *Neural computation*, vol. 18, no. 7, pp. 1527–1554, 2006.

[5] Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle, "Greedy layer-wise training of deep networks," *Advances in neural information processing systems*, vol. 19, pp. 153, 2007.

[6] M.A. Ranzato, F.J. Huang, Y.L. Boureau, and Y. LeCun, "Unsupervised learning of invariant feature hierarchies with applications to object recognition," in *Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on*. IEEE, 2007, pp. 1–8.

[7] A. Waibel, T. Hanazawa, G. Hinton, K. Shikano, and K.J. Lang, "Phoneme recognition using time-delay neural networks," *Acoustics, Speech and Signal Processing, IEEE Transactions on*, vol. 37, no. 3, pp. 328–339, 1989.

[8] R.P. Lippmann, "Review of neural networks for speech recognition," *Neural computation*, vol. 1, no. 1, pp. 1–38, 1989.

[9] G. Hinton, L. Deng, D. Yu, G. Dahl, A. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. Sainath, et al., "Deep neural networks for acoustic modeling in speech recognition," *IEEE Signal Processing Magazine*, 2012.

[10] F. Seide, G. Li, and D. Yu, "Conversational speech transcription using context-dependent deep neural networks," in *Proc. Interspeech*, 2011, pp. 437–440.

[11] G.E. Dahl, D. Yu, L. Deng, and A. Acero, "Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition," *Audio, Speech, and Language Processing, IEEE Transactions on*, vol. 20, no. 1, pp. 30–42, 2012.

[12] D. Yu and M.L. Seltzer, "Improved bottleneck features using pretrained deep neural networks," *Proc. Interspeech 2011*, pp. 237–240, 2011.

[13] A. Mohamed, G. Hinton, and G. Penn, "Understanding how deep belief networks perform acoustic modelling," in *Acoustics, Speech and Signal Processing (ICASSP), 2012 IEEE International Conference on*. IEEE, 2012, pp. 4273–4276.

[14] L. Mangu, H.K. Kuo, S. Chu, B. Kingsbury, G. Saon, H. Soltau, and F. Biadsy, "The IBM 2011 GALE Arabic speech transcription system," in *Automatic Speech Recognition and Understanding (ASRU), 2011 IEEE Workshop on*. IEEE, 2011, pp. 272–277.

[15] T.N. Sainath, B. Kingsbury, and B. Ramabhadran, "Auto-encoder bottleneck features using deep belief networks," in *Acoustics, Speech and Signal Processing (ICASSP), 2012 IEEE International Conference on*. IEEE, 2012, pp. 4153–4156.

[16] P. Vincent, H. Larochelle, Y. Bengio, and P.A. Manzagol, "Extracting and composing robust features with denoising autoencoders," in *Proceedings of the 25th international conference on Machine learning*. ACM, 2008, pp. 1096–1103.

[17] M. Finke, P. Geutner, H. Hild, T. Kemp, K. Ries, and M. Westphal, "The Karlsruhe-Verbmobil speech recognition engine," in *Acoustics, Speech, and Signal Processing, 1997. ICASSP-97., 1997 IEEE International Conference on*. IEEE, 1997, vol. 1, pp. 83–86.

[18] H. Soltau, F. Metze, C. Fugen, and A. Waibel, "A one-pass decoder based on polymorphic linguistic context assignment," in *Automatic Speech Recognition and Understanding, 2001. ASRU'01. IEEE Workshop on*. IEEE, 2001, pp. 214–217.

[19] T. Schaaf and F. Metze, "Analysis of gender normalization using MLP and VTLN features," in *Proc. Interspeech*, 2010.

[20] James Bergstra, Olivier Breuleux, Frédéric Bastien, Pascal Lamblin, Razvan Pascanu, Guillaume Desjardins, Joseph Turian, David Warde-Farley, and Yoshua Bengio, "Theano: a CPU and GPU Math Expression Compiler," in *Proceedings of the Python for Scientific Computing Conference (SciPy)*, June 2010, Oral Presentation.

[21] "IARPA, Office for Incisive Analysis, Babel Program," `http://www.iarpa.gov/Programs/ia/Babel/babel.html`, Retrieved 2013-03-06.

[22] J.J. Godfrey, E.C. Holliman, and J. McDaniel, "SWITCHBOARD: Telephone speech corpus for research and development," in *Acoustics, Speech, and Signal Processing, 1992. ICASSP-92., 1992 IEEE International Conference on*. IEEE, 1992, vol. 1, pp. 517–520.