Odyssey 2014:
The Speaker and Language Recognition Workshop
16-19 June 2014, Joensuu, Finland

Neural Network Bottleneck Features for Language Identification

Pavel Matejka[1,2], *Le Zhang*[1], *Tim Ng*[1], *Sri Harish Mallidi*[3]
Ondrej Glembek[1,2], *Jeff Ma*[1], *Bing Zhang*[1]

[1]Raytheon BBN Technologies, USA
[2]Brno University of Technology, Czech Republic
[3]The John Hopkins University, USA
{pmatejka,lzhang,tng,oglembek,jma,bzhang}@bbn.com

Abstract

This paper presents the application of Neural Network Bottleneck (BN) features in Language Identification (LID). BN features are generally used for Large Vocabulary Speech Recognition in conjunction with conventional acoustic features, such as MFCC or PLP. We compare the BN features to several common types of acoustic features used in the state-of-the-art LID systems. The test set is from DARPA RATS (Robust Automatic Transcription of Speech) program, which seeks to advance state-of-the-art detection capabilities on audio from highly degraded radio communication channels. On this type of noisy data, we show that in average, the BN features provide a 45% relative improvement in the C_{avg}or Equal Error Rate (EER) metrics across several test duration conditions, with respect to our single best acoustic features.

Index Terms: language identification, noisy speech, robust feature extraction

1. Introduction

The goal of the DARPA RATS (Robust Automatic Transcription of Speech) program is to create technology capable of accurately determining speech activity regions, detecting key words, and identifying language and speakers in highly degraded, weak and/or noisy communication channels. The Patrol team, led by BBN, participates in all the RATS tasks. The LID system that the Patrol team built for the RATS Phase 1 evaluation was described in [1]. That paper mainly focused on the individual systems built by members of the Patrol team, as well as on the calibration and the fusion of the individual systems. The Patrol team single best system for RATS Phase 2 evaluation was summarized in [2].

The neural network (NN) based features become an inseparable part of present-day state-of-the-art Large Vocabulary Continuous Speech Recognition (LVCSR) systems [3] but according to our best knowledge, there has not been any effort in applying it to conventional acoustic LID. This paper describes the usage of bottleneck (BN) features in the context of Language Identification (LID).

A BN feature of a given frame of audio can be interpreted as a compression of the information about the frame's phonetic class (and its phonetic context)—given as vector of (context-dependent) phoneme posterior—into a low dimensional vector. There were previous attempts of using phoneme-based features for frame-by-frame acoustic-based system. Ma et al. [2] used log of phoneme posteriors generated by neural network in conjunction with a block of PLP stream followed by HLDA dimensionality reduction and reports dramatic gain. Diez et al. [4] used phone log-likelihood ratios (PLLR) as an input to an acoustic-based system, and fused it with a MFCC-SDC system on the score level. Generally, both of these approaches share the same idea, only the fusion is done on a different level: feature- versus score-level. Han and Pelecanos [5] used a Arabic phone recognizer and applied Shifted Delta Cepstra [18] concept to capture linguistic information. The second approach they compared was to stack several frames of phoneme posteriors with PCA dimensionality reduction. They report nice gain on 120sec condition on RATS task. Similar procedure and analysis is done by Wang [6] on NIST LRE 2005 task.

At the time of concluding this work we found out that similar work with bottleneck features has been done independently by Song et.al [7] on NIST LRE 2009 task.

Conventional Phoneme Recognition followed by Language Model (PRLM) systems (generally called phonotactic system) usually build phoneme lattices from phoneme posteriors and derive expected trigram counts from these lattice. Such counts are then modeled with language-modeling techniques, Support Vector Machine [8], phonotactic i-vector extraction based on Subspace Multinomial Model (SMM) [9]. In the phonotactic system, it was shown, that at least trigrams need to be used to perform well. We believe that the key point is to use context dependent phoneme as targets in acoustic based LID. We propose to use the bottleneck features which compress the context dependent phonemes into low dimensional vector using bottleneck in the Neural Network. Similar to the phonotactic system our BN features tries to encode information about the phonetic context, but they can also take an advantage of machinery of acoustic system which was built over time and proved to be very good. It would be difficult to use context dependent phoneme posteriors in Mireia Diez's and Jeff Ma's approaches mentioned above due to the high dimensional output of posteriors from NN.

2. Stacked Bottleneck Features (SBN)

Bottleneck Neural-Network (BN-NN) refers to such topology of a NN, one of whose hidden layers has significantly lower dimensionality than the surrounding layers. It is assumed that such layer—referred to as the bottleneck—compresses the information needed for mapping the NN input to the NN output, increasing the system robustness to noise and overfitting. A bottleneck feature vector is generally understood as a by-product of forwarding a primary input feature vector through the BN-NN and reading off the vector of values at the bottleneck layer. In other words, after a BN-NN is trained for its primary task, the bottleneck layer is declared to be the output layer and all suc-

ceeding layers are ignored. Such NN then maps the primary features to the bottleneck features.

We have used a cascade of two such NNs. The output of the first network is *stacked* in time, defining context-dependent input features for the second NN, hence the term Stacked Bottleneck Features.

2.1. SBN Input Feature Extraction

Frequency domain linear prediction (FDLP) is an efficient technique to obtain a smooth parametric model of temporal envelope [10, 11]. Long segments of input speech (of the order of 10 seconds) are transformed into frequency domain using discrete cosine transform (DCT). The DCT samples are decomposed into sub-band DCT coefficients by applying critical band windowing (Bark). The sub-band temporal envelopes are then computed by applying FDLP on the sub-band DCT samples. The envelopes are compressed using a static compression scheme, which is a logarithmic function and dynamic compression scheme [12]. The logarithmic compression is to model the overall non-linear compression in the auditory system. The transitions are enhanced by the dynamic compression. The first part of Figure 1 shows the proposed feature extraction technique. The compressed envelopes are divided into 200 ms segments with a shift of 10 ms. DCT is applied both on static and dynamic compressed envelopes to obtain modulation spectrum representation. We use 14 modulation frequency components from each cosine transform, to cover modulation range of 0-35 Hz, resulting in 28 coefficients per band.

Although it is in 16KHz sampling rate, the RATS audio data is originally from the telephone corpora. Configurations for narrow-band data in signal processing are used. The lower cut-off frequency and the higher cut-off frequency are set to 125Hz and 3800Hz, respectively. In this frequency region, 17 critical bands are obtained (in Bark scale). This results in 476 modulation coefficients (28 coefficients per band). In addition to the 476 FDLP features, a pitch value is estimated for each frame using the RAPT algorithm [13] followed by a speaker-based mean and variance normalization. We expand the pitch feature context with an 11-frame concatenation. These 11 pitch values are then appended to the 476 FDLP features and the resulting 487 features are input to the NN training.

2.2. Neural Network Architecture

For the NN training, stacking strategy of a cascade of Neural Networks is shown in Figure 1 and described in details in [3, 14]. The configuration for the first NN is 487x1500x1500x80x1500xN, where *N* is the number of targets. The 80 bottleneck outputs from the first NN are sampled at times t, $t-10$, $t-5$, $t+5$ and $t+10$. Where t is the index of the current frame. The resulting 400-dimensional features are input to the second NN with a configuration of 400x1500x1500x80x1500xN. The bottleneck layers in both NNs have linear activation function which was shown to provide better performance [15]. All other hidden layers have sigmoid as the non-linearity. The 80 bottleneck outputs from the second NN are taken as features for conventional GMM–UBM–i-vector based LID system. The targets for training both NN are context-dependent cross-word quinphone codebooks which are taken from a LVCSR PLP system using the state-clustering approach as described in [16]. We used BBN Neural Network software to train NN with Stochastic gradient descent algorithm and batch size 512. The software uses the Graphics Processing Unit (GPU) for faster training. The weights are initialized with Gaussian distribution with zero mean and unity variance and biases are initialized with uniform distribution in the range $(-4.1, -3.9)$. The network is trained in about 15 iterations and for Farsi it reaches 23% frame level accuracy for the first NN and 28% for the second NN. We used two languages to train Stacked bottleneck (SBN) architecture, Levantine where number of targets is N=3707 and Farsi where N is 5306. The data used is derived from the RATS keyword search training corpus, which consists of retransmitted telephone data (CallFriend, Fisher and other RATS telephone data) over 8 radio channels. The amount of training data is 480 hours for Levantine and 350 hours for Farsi.

3. The RATS LID Data Corpus

The Linguistic Data Consortium (LDC) provided training and test data for the RATS evaluation tasks. For the LID evaluation task, the provided audio recordings cover 5 target languages (Arabic, Dari, Farsi, Pushtu, and Urdu) and 10 non-target languages (English, Spanish, Mandarin, Thai, Vietnamese, Russian, Japanese, Bengali, Korean, Tagalog). These recordings were selected from both existing data resources and new data collected specifically for RATS (more details can be found in [1]). All recordings were about 2 minutes long and were retransmitted through 8 different communication channels, labeled by the letters A through H. The retransmitted data was released to the RATS participants for developing their evaluation systems. LDC issued 3 incremental data releases for the LID task: LDC2011E95, LDC2011E111, and LDC2012E03. We only used the first two releases for developing our LID systems. All LID systems were evaluated under four testing conditions in which test samples are 120, 30, 10 and 3 seconds long, respectively. The RATS program does not provide development data for the short-duration conditions, 30s, 10s and 3s. Hence the participants need to find ways to develop systems for the short-duration conditions. As described in [8], we partitioned the first two data releases into training and development sets and created the 30s, 10s and 3s short cuts from the 120s audio files.

The total number of test samples in the development is about 7,120 samples for each condition. In the rest of the paper, we use Dev to denote this development set. We also measured LID performance on one adjudicated version of the LID Phase 1 evaluation data (also called Dev2 within the RATS program), which includes 1,914, 1,782, 1,715, 1,340 samples for the 120s, 30s, 10s and 3s conditions, respectively. We use Eval to denote this evaluation set.

4. LID System Description

First, we briefly review the LID system, that we use as our baseline. The BBN LID system consisted of 4 major components, speech activity detection (SAD), feature extraction, i-vector estimation, and neural network (NN) LID classifier.

4.1. Speech Activity Detection (SAD)

SAD system was carried out in three steps [17]. First, the input frame-level acoustic features (PLP) were projected to a lower-dimensional space using heteroscedastic linear discriminant analysis (HLDA). There are two classes for HLDA: speech and non-speech. Second, the reduced features were used to compute per-frame log likelihood scores with respect to speech and non-speech classes, each class being represented separately by 2048 Gaussian mixture model (GMM). Third, the frame-

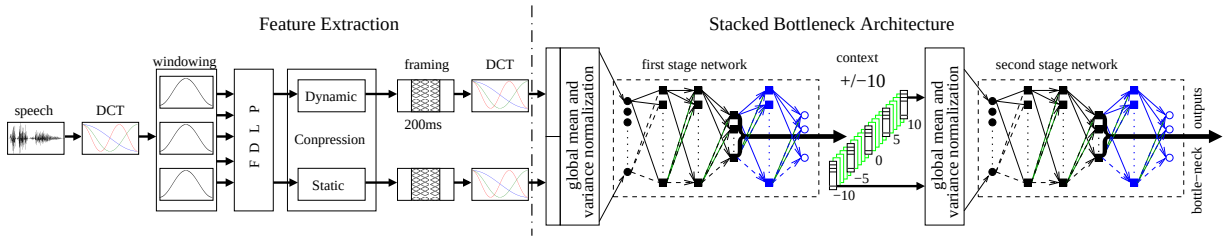

Figure 1: Block diagram of Stacked Bottleneck feature extraction with FDLP front-end.

level log likelihood scores were mapped to speech/non-speech classification decisions to produce final segmentation outputs. The mapping was done by thresholding the average per-frame log likelihood ratios, computed over the sliding window.

4.2. Baseline Feature Extraction

We found in [2] that, in terms of LID performance, it was best to project 11 frames, each including energy and the first 8 PLP coefficients, down to 60-dimensional feature vector by HLDA (we used languages as classes in the HLDA). We also found, this configuration produced better LID performance than the SDC coefficients [18]. We used Short Time Gaussianization (STG) [19] prior to HLDA dimensionality reduction, which further increased the robustness of the system.

4.3. i-vector Estimation

I-vector systems provide an elegant way of reducing the large-dimensional variable-length input data (time sequence of features) to a small-fixed-dimensional feature vector—referred to as the *i-vector*—while retaining most of the relevant information [20, 21]. The technique was originally inspired by Joint Factor Analysis framework introduced in [22].

Our i-vector extractor was trained in 10 iterations of jointly applying the Expectation Maximization (EM) algorithm and the Minimum Divergence (MD) step [23]. Sufficient statistics for both the i-vector extractor training and the i-vector estimation were collected using a 1024-component GMM. The i-vector dimensionality was set to 400.

As mentioned before, all training samples have at least 2 minutes of audio. So, in terms of audio length these training samples do not match the short-duration (30s, 10s and 3) test samples. To reflect the shorter duration test conditions during training, we estimated the 3s i-vectors for audio chunks of approximately 3 seconds of speech. The chunks were generated by grouping adjacent speech regions within each training audio file. We then combined these 3s i-vectors with the regular i-vectors estimated on the entire audios to train the final LID classifier. We experimented with different windows and overlaps for cutting and generating i-vectors using 3s cuts provided the best performance as shown in [2]. In our previous experiments [1], we showed that adding these short i-vectors help only the NN classifier [1] but not Logistic Regression (LR) classifier.

4.4. NN LID Classifiers

In [1], we have shown that Neural network (NN) classifier significantly outperforms Logistic Regression. We have compared NNs to Logistic regression, Gaussian backend, Cosine distance and also recently introduced Discriminative Adaptive Gaussian Backend [24] for Phase-3 evaluation and NN gives us consistently the best performance across many systems. We used the ICSI Quicknet NN tools to train NNs to map i-vectors to language posteriors. We configured all our NNs with 3 layers (input, hidden, and output) with the input layer taking the i-vectors and the output layer generating posteriors for the 6 language classes. We trained 5 NNs with the number of hidden nodes set to 300, 400, 500, 600 and 700. We adopted this approach to increase robustness, accuracy and to avoid the over fitting problem. We can see the analogy in machine learning where it is called bootstrap aggregating (bagging). Similar results can be obtained when we train 5 NNs with 400 hidden nodes and different random initialization of the weights. We trained separate Logistic Regression (LR) classifier on our development set to calibrate all 5 NN outputs. Then we take average of posterior probability from LR. We take log of the average posterior as final score of the system.

5. Results

5.1. Evaluation Metric

In accordance with the RATS program targets we focused on improving the LID system performance on the 10s and 3s test conditions. We will report the performance measured on these two short-duration conditions in this paper. The final table will show also the results for 30sec condition. We measured the LID performance according to four metrics, Acc (accuracy defined as correctly recognized samples divided by all samples), EER (we first compute EER for each language and then take average), C_{avg}(computed the same way as in the NIST LRE evaluation [25]), and one of the RATS Phase-3 operating points: miss rate at false alarm rate equal to 1% (we denote it as $P_{miss@FA1\%}$). We report the Acc, EER, C_{avg}, $P_{miss@FA1\%}$ scores as percentage numbers in this paper on the Dev data. Last table shows the results also on the eval data.

5.2. Baseline Experiments

We ran several common acoustic features used nowadays in Language Identification for comparison with our SBN features. The short description of the baseline features follow:

PLP: We extract 8 PLP coefficients plus normalized energy using a 25ms Hamming window with a 10ms frame shift.

MFCC: This front-end operates on standard Mel-frequency Cepstrum Coefficients (MFCC), extracted using a 25ms Hamming window. We extract 8 MFCCs together with C0 every 10ms.

MHEC: The Hilbert envelope is calculated on the Gammatone filter bank and smoothed with low-pass filter with cut-off frequency of 20Hz. Framing with window 25ms and 10ms shift is applied. Long term normalization is used before taking logarithm and DCT. This results in 9 dimensional features including C0. For more details see [27].

PLP2: The output power spectral estimates from the critical band integration stage of FDLP(see Section 2.1), are inverse

Table 1: *Comparison of different acoustic features with phoneme based features. Asterisk "*" denotes that, instead of stacking multiple frames and applying HLDA projection, only one feature frame is used in LID modeling. All numbers are on Dev set in [%]*

	3 sec			10 sec		
	Acc	C_{avg}	EER	Acc	C_{avg}	EER
MHEC	64.82	17.45	16.91	82.95	7.76	7.39
PLP	65.90	17.10	16.75	82.85	8.22	7.88
MFCC	60.10	20.01	19.63	78.78	9.57	9.32
PLP2	61.33	18.85	18.40	80.66	8.82	8.25
logP*	69.31	16.15	15.51	85.52	7.14	6.96
logP	71.34	14.77	14.13	87.88	6.05	5.85
BN*	71.93	13.68	13.37	89.70	4.70	4.44
SBN*	72.78	13.21	12.70	90.37	4.26	4.10

Fourier transformed to obtain an autocorrelation sequence. This sequence is used for time-domain linear prediction (TDLP), using a 19th-order model. The TDLP provides an all-pole approximation of the short-term spectrum. The output TDLP parameters are converted to 9 cepstral coefficients using cepstral recursion [28].

logP*: The log phoneme posteriors are also very strong features when modeled with acoustic system. We trained one neural network to produce 39 context independent phonemes for Levantine. The input to the NN is a block of 9 frames of 12 PLP plus energy, deltas and double-deltas making together 351 dimensional vector. The neural network has 3 hidden layers with 1500 neurons and output layer with 39 neurons. Only one frame of 39 dimensional log-posterior vector is taken as an input to the LID system. The system significantly outperformed any acoustic feature-based system presented in Table 1.

logP in Table 1 presents results when the input to the LID system is a stack of 11 frames of logP* concatenated together and projected via HLDA (with language labels defining the classes) to the 60 dimensional feature vector. Better results suggest that there is an advantage to take the context into account.

BN*, SBN*: Last two lines of Table 1 present the results for our best bottleneck features trained on Farsi with context dependent phonemes as targets. BN* is 80 dimensional bottleneck from the first stage Neural Network (see Figure 1) and SBN* is 80 dimensional bottleneck from the second stage NN in our cascade. Only one frame of these features are fed into LID system. The relative improvement in $C_{avg}$is 25% for 3sec condition, 45% for 10sec condition and 60% for 30sec condition (see also Table 5) over the best MHEC or PLP features.

The procedure of feature extraction, post-processing and, normalization for PLP, PLP2, MFCC and MHEC are described in Section 4.2.

5.3. Different Targets in SBN

One of the main question is: What target should we train the SBN features on? With bottleneck features we "do not care" what the final targets are, because we take bottleneck from the NN and ignore the succeeding layers. With this approach, we can afford to use context dependent phonemes, as they contain more context information and in this sense are closer to the phonotactic system where trigrams are usually used. The answer is in Table 2. We trained the same topology of the SBN with only one difference. The targets for both NN in the SBN were trained with context independent (CI) phonemes in the first case and context dependent phonemes (CD) in the second case. There is significant improvement mainly for the longer duration files with using CD phonemes.

Table 2: *Comparison of different targets in training of SBN features. The number of context independent targets—CI is 49 and context dependent targets—CD is 5306. The numbers in brackets are number of targets. Both systems use 80 dimensional SBN features. All numbers are on Dev set in [%]*

	3 sec			10 sec		
	Acc	C_{avg}	EER	Acc	C_{avg}	EER
CI	70.33	14.46	14.22	88.79	5.30	5.16
CD	72.78	13.21	12.70	90.37	4.26	4.10

Table 3: *Comparison of different size of the SBN features used in the second NN. All numbers are on Dev set in [%]*

	3 sec			10 sec		
	Acc	C_{avg}	EER	Acc	C_{avg}	EER
20	66.64	17.05	16.71	85.31	6.75	6.49
40	69.57	14.98	14.59	88.62	5.44	5.10
60	70.86	14.03	13.69	90.07	4.77	4.43
80	72.78	13.21	12.70	90.37	4.26	4.10
100	71.88	13.36	13.06	90.69	4.23	4.09
120	71.23	13.70	13.16	90.84	4.21	4.02

5.4. Size of Bottleneck Layer

We investigated also in the size of the bottleneck in the second NN which directly influence the size of the LID system. The results in Table 3 shows that the performance start to saturate above 60 with optimum at 80.

5.5. Language Dependent System

Based on the analysis in [2], we also ran the "Language-Dependent" (LD) system, where 6 systems with different language-dependent Universal Background Model (UBM) were trained and fused together to form the final score. In our system, we have 6 classes - 5 for target languages and one for the non-target languages. Together, we trained 6 UBMs only on the data belonging to the one language class. The i-vector extractor and the NN classifier were trained on the data from all languages. The final system is an average of language posterior probabilities from final NN classifier of separate language dependent systems. Note that the separate systems are first calibrated using Logistic regression, as described in Section 4.4. Table 4 presents the results for the "baseline" language independent (LI) system, with a single UBM trained on all data. Next, there are 6 separate LD systems, where first column of Table 4 denotes the language on which the UBM was trained. There is There is a small degradation in performance against baseline. This is expected since the UBM was trained only on one language hence does not generalize good on other languages. The best LD system is when we train the UBM on 10 out of set class languages

Table 5: *Final comparison of baseline PLP and Stacked bottleneck features on Dev and Eval data for 3s, 10s and 30s conditions. All numbers are in [%].* $\mathrm{P_{miss}}$ *denotes miss rate at false alarm rate equal to 1%.*

		3 sec				10 sec				30 sec			
Dev set		Acc	$\mathrm{C_{avg}}$	EER	$\mathrm{P_{miss}}$	Acc	$\mathrm{C_{avg}}$	EER	$\mathrm{P_{miss}}$	Acc	$\mathrm{C_{avg}}$	EER	$\mathrm{P_{miss}}$
1	PLP	65.90	17.10	16.75	61.76	82.85	8.22	7.88	26.45	90.65	4.20	3.94	11.48
2	SBN FAR	72.78	13.21	12.70	49.37	90.37	4.26	4.10	12.84	96.65	1.61	1.47	2.10
3	SBN LEV	72.97	13.10	12.88	48.89	91.51	4.15	3.98	10.61	96.50	1.56	1.55	2.10
4	SBN LDFAR	77.33	11.26	10.85	42.40	93.10	3.22	3.36	7.81	97.24	1.27	1.21	1.38
5	SBN LDLEV	78.01	10.82	10.53	39.47	93.50	3.09	3.08	6.87	97.24	1.15	1.09	1.14
6	AVG(4+5)	80.27	9.64	9.59	34.92	94.43	2.71	2.66	5.32	97.73	1.05	1.00	0.92
Eval set		Acc	$\mathrm{C_{avg}}$	EER	$\mathrm{P_{miss}}$	Acc	$\mathrm{C_{avg}}$	EER	$\mathrm{P_{miss}}$	Acc	$\mathrm{C_{avg}}$	EER	$\mathrm{P_{miss}}$
1	PLP	61.84	17.99	18.25	60.10	75.80	13.34	12.95	34.73	82.49	9.75	10.32	19.43
2	SBN FAR	68.71	14.42	13.72	46.54	84.55	9.38	6.84	21.30	90.97	5.89	4.65	8.52
3	SBN LEV	71.17	13.97	13.81	42.89	83.91	9.47	6.29	21.39	91.92	5.84	4.81	8.96
4	SBN LDFAR	73.26	13.43	12.36	38.59	87.58	6.44	5.13	15.45	92.76	5.11	4.06	5.86
5	SBN LDLEV	74.91	12.88	12.38	35.07	87.52	8.09	4.71	14.08	92.93	5.81	3.96	5.68
6	AVG(4+5)	76.03	11.73	10.83	32.72	89.10	6.50	4.25	12.07	93.77	4.94	3.53	5.15

Table 4: *Analysis of language dependent system. Comparison of separate systems where UBM is trained only on one language. Done with Farsi SBN80 features. All numbers are on Dev set in [%]*

	3 sec			10 sec		
	Acc	$\mathrm{C_{avg}}$	EER	Acc	$\mathrm{C_{avg}}$	EER
baseline	72.78	13.21	12.70	90.37	4.26	4.10
alv	72.04	13.46	13.11	90.12	4.54	4.37
fas	72.01	13.24	12.63	90.58	4.40	4.21
prs	71.80	12.64	12.39	90.32	4.43	4.30
pus	71.00	13.70	13.27	90.13	4.43	4.43
urd	71.88	13.74	13.45	89.73	4.81	4.72
xxx	73.95	13.81	13.03	91.03	4.37	4.20
avgLang	77.33	11.26	10.85	93.10	3.22	3.36
avgChan	78.05	10.58	10.40	93.37	3.32	3.16
avgAll	78.21	10.48	10.23	93.55	3.14	3.20

marked as "xxx". This was also expected because the UBM is more general since it was trained on multiple languages. Still, it does not reach the performance of the LI system, because it was not trained to model the target languages. Finally, "avgLang" represents the average of scores of the 6 separate LD systems. The relative gain is 20% for 3sec condition and 15% for 10sec condition over the LI system.

We have nine channels in RATS program. If we repeat the same experiment with channel dependent system and train the UBM for every channel and we get similar results ("avgChan" in Table 4). If we average all channel- and language-dependent systems, we get about the same results ("avgAll" in Table 4).

5.6. Different target language in training of SBN

There are two languages in RATS program that we have word transcriptions for which we have built speech to the text system for the keyword spotting task. We have built Farsi and Levantine SBN features and tested it for LID. Lines 2 and 3 in Table 5 shows the results of the SBN LID system where we used different language for training the SBN features. The results are very comparable with small favor in Levantine system.

5.7. Fusion

The final summary is in Table 5. In this table, we show in addition also one of the RATS Program target metrics $\mathrm{P_{miss@FA1\%}}$, we also report results on the 30sec condition (120s is too good to report anything on) and also full results on the evaluation data. First system presents PLP baseline system. Next two systems (2 and 3) show the two stack-bottleneck systems trained on Farsi and Levantine. Next two systems (4 and 5) are language dependent variants of the 2 and 3. System number 6 is the fusion (average of the scores) of systems 4 and 5. This fusion gives in average 10% relative improvement on the development and also evaluation data.

6. Conclusion

We have presented the bottleneck features in the context of Language identification. It combines benefits of both phonotactic and acoustic system. Usually, the phonotactic system is favorable for the long duration files, while acoustic for the short ones. This approach takes the advantage of both. In addition, we can also use modeling of context dependent phonemes in bottleneck features. This brings very nice improvement over the context independent phonemes.

Overall the bottleneck features provide dramatic relative improvement 25% for 3sec condition, 45% for 10sec condition and 60% for 30sec condition for $\mathrm{C_{avg}}$. The same or higher relative gains can be seen also for the primary RATS metric $\mathrm{P_{miss@FA1\%}}$ or average EER. We can extend the basic variant of the LID system to Language dependent system where UBM is trained on the subset of the data. This technique gives us additional 15% relative improvement. This system also fuse well with both classic acoustic and phonotacic approaches.

7. Acknowledgments

This work was supported by the DARPA RATS Program. The views expressed are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government

8. References

[1] Matejka, P., et al., ”Patrol Team Language Identification System for DARPA RATS P1 Evaluation”, Interspeech 2012, Portland, Oregon, USA, 2012.

[2] Ma, J. et al., ”Improvements in Language Identification on the RATS Noisy Speech Corpus”, Interspeech 2013, Lyon, France, 2013.

[3] Grezl, F., Karafiat, M., Burget, L.: ”Investigation into Bottleneck Features for Meeting Speech Recognition”, Interspeech 2009, Brighton, GB, 2009.

[4] Diez, M., Varona, A., Penagarikano, M., Rodriguez-Fuentes, L.J., Bordel, G., ”On the use of Phone Log-likelihood Ratios as Features in Spoken Language Recognition”, SLT 2012, Miami, Florida USA, 2012.

[5] Han, K.J. and Pelecanos, J., ”Frame-based Phonotactic Language Identification”, Spoken Language Technology Workshop (SLT), 2012 IEEE.

[6] Wang, H., Leung, Ch., Lee, T., Ma, B., Li, H., ”Shifted-Delta MLP Features for Spoken Language Recognition”, Signal Processing Letters, IEEE (Volume:20 , Issue: 1), pp. 15–18, 2013.

[7] Y. Song et al, ”i-vector Representation Based on Bottle Neck Feature for Language Identification”, IEEE Electronics Letters, 2013.

[8] Campbell, W.M. and Singer, E. andTorres-Carrasquillo, P.A. and Reynolds, D.A., ”Language Recognition with Support Vector Machines”, Speaker Odyssey 2004, Toledo, Spain, 2004,

[9] Soufifar, M., Burget, L., Plchot, O., Cumani, S., ernock, J.: ”Regularized Subspace n-Gram Model for Phonotactic i-vector Extraction”, Interspeech 2013, Lyon, FR, 2013.

[10] Athineos, M. and Ellis, D., ”Autoregressive Modelling of Temporal Envelopes,” IEEE Tran. Signal Proc., Vol. 55, pp. 5237-5245, 2007.

[11] Ganapathy, S., Thomas, S., and Hermansky, H., ”Static and Dynamic Modulation Spectrum for Speech Recognition”, Interspeech 2009, Brighton, UK, Sept. 2009.

[12] Tchorz, J., and Kollmeier, B., ”A Model of Auditory Perception as Front End for Automatic Speech Recognition”, J. Acoust. Soc. Am., Vol. 106(4), pp. 2040-2050, 1999.

[13] Talkin, D., ”A Robust Algorithm for Pitch Tracking (RAPT), in Speech Coding and Synthesis” , W. B. Kleijn and K. Paliwal, Eds. New York: Elseviever, 1995.

[14] Grézl, F., Karafiát, M., and Veselý, K., ”Adaptation of Neural Network Feature Extractor for New Language”, Interpeech 2013, Lyon, France, 2013.

[15] Veselý, K., Karafiát, M., and Grézl, F., ”Convolutive Bottleneck Network Features for LVCSR”, Proceedings of ASRU 2011, IEEE Signal Processing Society, Hawaii, USA, 2011.

[16] Matsoukas, S., et al., ”Advances in Transcription of Broadcast News and Conversational Telephone Speech Within the Combined EARS BBN/LIMSI System”, IEEE Transactions on Audio, Speech, and Language Processing, vol. 14, no. 5, pp. 1541-1556, Sep 2006.

[17] Ng, T., Zhang, B., Nguyen, L., et al., ”Developing a Speech Activity Detection System for the DARPA RATS Program”, Interspeech 2012, Portland, Oregon, USA, 2012.

[18] Torres-Carrasquillo, P. A., Singer, E., et al., ”Approaches to Language Identification Using Gaussian Mixture Models and Shifted Delta Cepstral Features”, ICSLP 2002, Denver, Colorado, USA, 2002.

[19] Xiang, B., Chaudhari, U.V., Navrátil, J., Ramaswamy, G.N., Gopinath, R.A., ”Short-time Gaussianization for Robust Speaker Verification”, ICASSP 2002, Orlando, Florida, USA, May 2002.

[20] Dehak, N., Kenny, P., Dehak, R., Dumouchel, P. and Ouellet, P., ”Front-end Factor Analysis for Speaker Verification”, IEEE Transactions on Speech and Audio Processing, Vol. 19, No.4, 2011.

[21] Glembek, O., Burget, L., Matejka, P., Karaiat, M. and Kenny, P., ”Simplification and Optimization of i-vector Extraction”, ICASSP 2011,Prague, Czech Republic, May 2011.

[22] P. Kenny, ”Joint Factor Analysis of Speaker and Session Variability : Theory and Algorithms”, Technical report CRIM-06/08-13, Montreal, CRIM, 2005.

[23] Niko Brummer, ”The EM Algorithm and Minimum Divergence”, Agnitio Labs Technical Report. Online: http://niko.brummer.googlepages.com/EMandMINDIV.pdf, Jan. 2014.

[24] McLaren, M., Lawson, A., Lei, Y., Scheffer, N., ”Adaptive Gaussian Backend for Robust Language Identification”, Interspeech 2013, Lyon, France, 2013.

[25] ”The 2009 NIST Language Recognition Evaluation Plan (LRE2009)”, Online: http://www.itl.nist.gov/iad/mig/tests/lre/ /2009/LRE09_EvalPlan_v6.pdf

[26] Li, Q., Huang, Y., ”Robust Speaker Identification Using an Auditory-based Feature”, ICASSP 2010, Dallas, TX, USA, March 2010.

[27] Sadjadi, S.O., Hansen, J.H.L., ”Hilbert Envelope Based Features for Robust Speaker Identification under Reverberant Mismatched Conditions”, ICASSP 2011 ,Prague, Czech Republic, May2011.

[28] Athineos, M., Ellis, D., ”Autoregressive Modelling of Temporal Envelopes”, IEEE Trans. of Signal Processing, vol. 55, pp. 5237–5245, 2007.