BUT-AGNITIO System Description for NIST Language Recognition Evaluation 2009

Niko Brümmer (*El Derivador*), Lukáš Burget (*El Discriminador*), Ondřej Glembek (*El Fonetista*), Valiantsina Hubeika (*La Chica*), Zdenek Jančík (*El Estadístico*), Martin Karafiát (*Zorro*), Pavel Matějka (*El Generador*), Tomáš Mikolov (*El Chico*), Oldřich Plchot (*El Organizador*), Albert Strasheim (*Pocoyó*)

Speech@FIT, Brno University of Technology, Czech Republic
and
AGNITIO, South Africa

1. Bugfix Update

All of our submitted systems were replaced with *bugfix* versions. The 'bug' was discovered after the results had been released and corrected subsequently. The corrected versions were re-submitted to NIST for scoring.

Only one subsystem, which happened to be included in all of our submissions, the one described in section 8.2, was subject to this error. Slightly different versions of this subsystem had been run on the development data and on the evaluation data. This turned out to be fatal to all our submissions.

In our bugfix submissions, we corrected only this error and left everything else as it was and as described below.

2. Introduction

BUT-AGNITIO are submitting 4 different systems, which we exercised on the *closed-set*, *open-set* and *language-pairs* tasks. Here we list these systems and specify which system should be considered as primary system for which task. Detailed descriptions follow later.

BUT-AGN-1: This is our primary system for *closed-set*. See section 6.1 for details.

BUT-AGN-2: This is our primary system for *open-set*. It was also exercised on *closed-set* as a contrastive system. See section 6.2 for details.

BUT-AGN-3: This is our primary system for *language-pairs*. It was also exercised on *closed-set* as a contrastive system. See section 6.3 for details.

BUT-AGN-4: This is a contrastive system for *closed-set*. See section 6.4 for details.

For all of our systems, the score in every trial may be interpreted as the *log-likelihood-ratio* for the target versus the non-target hypothesis.

3. Development data

The following data (distributed by LDC and ELRA) were used to train our systems.:

CF	CallFriend
F	Fisher English Part 1.and 2.
F	Fisher Levantine Arabic
F	HKUST Mandarin
SRE	Mixer (data from NIST SRE 2004,2005,2006, 2008)
LDC07	development data for NIST LRE 2007
OGI	OGI-multilingual
OGI22	OGI 22 languages
FAE	Foreigen Accented English
SpDat	SpeechDat-East [1].
SB	SwitchBoard
VOA	Voice of America radio broadcast

Our data was separated into two independent subsets, which we denoted TRAIN and DEV. The TRAIN subset had 54 languages (including the 23 target languages) and had about 80 000 segments in total. The DEV subset had 57 languages (including the 23 targets) and a total of about 60 000 segments. The DEV subset was split into balanced subsets having nominal durations of 3s, 10s and 30s. The DEV set was based on segments from previous evaluations plus additional segments extracted from longer files from CTS and VOA databases (which were not contained in the TRAIN set).

For a detailed breakdown of the amounts of training data per language, see Table 1 at the end of the document.

4. General System description

In this section we describe the general system architecture that is common to all systems. Each system has three main stages:

Frontends, of which there may be multiple different ones for a complete system. Each frontend stage maps the input speech segment to a *score-vector*. We denote these frontend outputs as *amorphous scores*. The dimensionality of these scores vary between 23 and 68, as described in more detail later.

Backend, which performs fusion and calibration. The backend fuses the amorphous scores from the frontends and outputs *calibrated scores*. These scores function as multiclass log-likelihoods. In the closed-set case there are 23 log-likelihoods per input segment, for each of the 23 target languages. In the Open-set case, there are 24: the 23 target log-likelihoods as well as the log-likelihood for the hypothesis that the input is from some other language. The backend is further described in the next section.

Decision stage, which takes the (i) backend output log-likelihoods and (ii) the priors as defined for each trial. These are used in Bayes' rule to obtain the posterior distribution over the language classes. The posterior is then used to make minimum-expected-cost Bayes decisions. For closed-set the prior allows 23 hypotheses, for open-set 24 hypotheses and for pairs 2 hypotheses. For each input segment, there are multiple detection trials, where the prior is varied between trials, as specified in the evaluation plan.

5. Backend

The backend maps one or more amorphous input score-vectors to a calibrated output score-vector, for every input segment. There are two backend variants, for closed-set and open-set respectively. Both variants are composed of separate Gaussian backends (GBE's) for different frontends, followed by a single discriminative fusion and calibration stage:

5.1. Gaussian Backend (GBE)

The GBE models the amorphous scores with a different Gaussian model in amorphous score-space, for each language class. In the closed-set case, all the class models share the same common within-class covariance (CWCC). In the open-set case, the 23 target languages share the CWCC, but the out-of-set class has a larger covariance. In all cases there are different class-conditional means.

For the closed-set case, we use maximum likelihood (ML) estimates for the parameters. The CWCC was estimated over all 57 languages, while we used the means only for the 23 target languages.

In the open-set case, we take the out-of-set covariance as CWCC+BCC, where BCC is the between-class covariance, estimated from the means of all 57 languages in DEV, so BCC was estimated from 57 data points. The mean for this model was chosen as the mean of the 57 language means.

The output scores of the GBE are the 23 or 24 log-likelihoods of the input score-vector, given each of the class models.

5.2. Fusion and calibration

In contrast to our previous work, where we used three separate backends for nominal durations of 3s, 10s and 30s, we built a single duration-compensated fusion and calibration stage this year.

Let there be M input systems, where system i produces amorphous score-vector $\mathbf{s}_{it}$ for a given input t. Each system also outputs as ancillary information an indication of the duration of the input segment, denoted d_{it}. For acoustic systems, this was the number of 10ms speech frames found by the VAD (voice-activity-detection). For phonotactic systems, this was the expected number of phones in the segment. Let $B()$ denote the mapping effected by the GBE, then the output of the fusion and calibration is:

$$\vec{\ell}_t = \sum_{i=1}^{N} a_{1i} B(\mathbf{s}_{it}) + a_{2i} B(d_{it}^{-0.5} \mathbf{s}_{it}) + a_{3i} B(d_{it}^{-1} \mathbf{s}_{it}) + \mathbf{b} + \mathbf{C}\vec{\gamma}_t \quad (1)$$

where a_{ji} are scalar fusion weights, $\mathbf{b}$ is an offset vector, $\mathbf{C}$ is a matrix and $\vec{\gamma}_t$ is a vector of ancillary data. For systems which fused both acoustic and phonotactic, we composed $\vec{\gamma}$ of the phone and frame durations, as well as their square roots. In cases where we fused more than one phonotactic system, we used the expected number of phones for each system.

Notice that for each system, we fused in three differently normalized score variants and for each of these variants, a different GBE was trained.

The fusion parameters $(a_{ji}, \mathbf{b}, \mathbf{C})$ were discriminatively trained using multiclass logistic regression [9, 10]. This tends to produce well-calibrated class log-likelihoods. We verified this fact by judging calibration on independent data (see jackknifing below), by comparing C_{avg} and C^*_{avg} (as defined below).

5.3. Jackknifing

We used our TRAIN data subset to train all frontends, while we used our DEV data to train all the backend stages and also to test performance. To keep backend training and test separate, we resorted to a jackknifing scheme. We did 5 outer iterations, where in each we randomly partitioned the DEV data into 5 subsets balanced across all 57 languages. In 5 inner iterations, one subset was held out as test data, while the other 4 were used for backend training.

We computed C_{avg} and C^*_{avg} on each of the 25 test sets and averaged. We also averaged the 25 backends thus obtained for a final backend which was applied to the LRE'09 evaluation data.

6. System compositions

Here we describe how the different systems are composed.

6.1. BUT-AGN-1

This is our primary system for *closed-set*. It is a selection of subsystems (frontends) which we judged to give the best performance. We made the final decision, based on C_{avg} for 10s duration, because we judged the error-rate on 30s duration was too low for reliable decisions.

This system fuses three acoustic systems and four phonotactic systems. The acoustic systems are:

- The RDLT system of section 8.2, *with* channel compensation as described in section 8.1.
- The RDLT system of section 8.2, with channel compensation omitted.
- The MMI system of section 8.3.

The phonotactic systems are:

- Three systems described in section 8.4, respectively using the Hungarian, Russian and English phone recognizers.
- The SVM system as described in section 8.5.

6.2. BUT-AGN-2

This is our primary system for *open-set*. It is almost the same BUT-AGN-1, with the following differences:

- The RDLT system without JFA channel compensation was omitted.
- The backend was trained in open-set mode.

It was also run on closed-set as a contrastive system, by simply ignoring the 24th log-likelihood output of the backend.

6.3. BUT-AGN-3

This is our primary system for *all-pairs*. It is almost the same BUT-AGN-1, with the following difference:

- The RDLT system without JFA channel compensation was omitted.

It was also run on closed-set as a contrastive system.

6.4. BUT-AGN-4

This is a monolithic system, composed of our single best acoustic system described in section 8.2. This is a contrastive system for closed-set.

7. Frontend types

There are two types of frontend, *acoustic* and *phonotactic*. Here, we give general descriptions of both types, followed by details of each frontend.

7.1. Acoustic

The acoustic systems are based on MFCC/SDC acoustic features. This is a brief summary of acoustic feature extraction and UBM training. For more detail, see our previous work [22, 7].

The inputs to the language recognizer are segments of recorded speech of varying duration. The voice activity detection (VAD) is performed by our Hungarian phoneme recognizer, with all the phoneme classes linked to 'speech' class.

All acoustic systems used the popular shifted-delta-cepstra (SDC) [3] feature extraction. The feature extraction is similar to BUT LRE 2005 system [2]. Every speech segment is mapped to a variable-length sequence of feature vectors as follows: After discarding silent portions, every 10ms speech-frame is mapped to a 56-dimensional feature vector. The feature vector is the concatenation of an SDC-7-1-3-7 vector and 7 MFCC coefficients (including C0). Cepstral mean and variance normalization are applied before SDC.

Vocal-tract length normalization (VTLN) performs simple speaker adaptation. We used MAP adaptation from UBM (single GMM with 32 diagonal Gaussians trained on Switchboard) to derive specific models for each warping factor [19]. Models are retrained using MMI (Maximum Mutual Information) criterion. The reference is derived by LVSCR system.

A 2048-component, language-independent, maximum-likelihood GMM was trained with the EM-algorithm on the pooled acoustic feature vectors of all 54 languages in the TRAIN dataset. We follow speaker recognition terminology and refer to this language-independent GMM as the *universal background model*, or UBM [8].

7.2. Phonotactic

The phonotactic systems were based on 3 phoneme recognizers: two left-context/right-context hybrids and one based on GMM/HMM context dependent models. All the recognizers are able to produce phoneme strings as well as phoneme lattices. In case of lattices, posterior-weighted counts ("soft-counts") were used in the following processing [15].

7.2.1. Hybrid phoneme recognizers

The phoneme recognizer is based on hybrid ANN/HMM approach, where artificial neural networks (ANN) are used to estimate posterior probabilities of phonemes from Mel filter bank log energies using the context of 310ms around the current frame. Hybrid recognizers were trained for Hungarian and Russian on the SpeechDat-E databases. For more details see [21, 13].

7.2.2. GMM/HMM phoneme recognizers

The third phoneme recognizer was based on GMM/HMM context dependent state clustered triphone models, which are trained in similar way as the models used in AMI/AMIDA LVCSR [14]. The models were trained using 2000 hours of English telephone conversational speech data from Fisher, Switchboard and CallHome databases. The features are 13 PLP coefficients augmented with their first, second and third derivatives projected into 39 dimensional space using HLDA transformation. The models are trained discriminatively using MPE criterion [18]. VTLN and MLLR adaptation is used for both training and recognition in SAT fashion. The triphones were used for phoneme recognition with a bi-gram phonotactic model trained on English-only data.

8. Frontend descriptions

This section lists details of all the different frontend variants.

8.1. JFA-2048G

This is an acoustic system, based on processing of sufficient statistics derived from the UBM, in the same way that Patrick Kenny does it for his 'JFA' speaker recognition systems [4, 5]. This system (JFA-2048G) is described in detail in [1].

This system does 'channel' compensation in the same way it is done for speaker recognition, via factor analysis of segment-dependent GMM models. The channel factor loading matrix $\mathbf{U}$, is trained via an EM algorithm over 500 sessions of each of the 23 target languages. The supervector dimensionality is about 10^5 and the dimensionality of the channel subspace is 200.

The language models are MAP-adapted with good-old relevance-MAP adaptation from the UBM [8]. For this system, we generated 68 models, to produce 68 frontend scores. We used all of the 54 available languages and trained two separate models for those languages that have both telephone and radio speech.

Scoring is done by language-independent channel-compensation, followed by linear scoring against each language model [6].

8.2. RDLT14L256-2048G

Region Dependent Linear Transforms (RDLT) [24] is a discriminatively trained feature extraction, which is generalization of a technique known in speech recognition as fMPE [25]. In our system, 256 linear transformations (56x56 matrices) take one common 56-dimensional feature vector of SDC+MFCC as an input. The outputs of the transformations are linearly combined to form a single 56-dimensional output feature vector. The mixing weights are given by posterior probabilities of 256 components of a GMM, which is trained on the same input features. The transformations are discriminatively trained in similar manner as described in [24, 25] to maximize the expected probability of a segment being correctly recognized using a set of language dependent GMMs, which are maximum likelihood trained on the RDLT output features. The average duration of training segments is about 1 second. After training the

RDLT, the set of language dependent GMMs is discarded, and the RDLT features are used to generate statistics for the JFA system described in 8.1.

8.3. MMI-FeaCC-2048G

This subsystem uses GMM models with 2048 Gaussians per language, where mean and variance parameters are re-estimated using Maximum Mutual Information criterion - the same as for BUT LRE2005 [2]. The SDC features are first compensated using eigen-channel adaptation in feature domain [23, 20]. Starting from target language models with means MAP adapted from UBM using the compensated features, mean and variance parameters are further re-estimated using MMI criterion [22].

8.4. EN-TREE-45-N4, HU-TREE-6133-N4, RU-TREE-50-N4

In all systems, binary decision tree language modeling was based on creating a single language independent tree (referenced as "UBM") and adapting its distributions to individual language training data, as described in Navratil's work [16, 17]. We used English, Hungarian, Russian phone recognizer to generate 4-gram lattice counts.

8.5. SVM-HU-N3

In this subsystem, the trigram-lattice-counts from Hungarian phoneme recognizer were used as features for subsequent classification by SVMs, similar to MIT's work [26].

9. Processing time

The development effort at the BUT lab in the last month alone, consumed 21.14 CPU years.

10. Evaluation criteria for development

The question of how to best judge the accuracy of language recognizers has been answered in the literature in different ways. A straight-forward solution is multiclass misclassification error-rate. However, this solution is lacking in two respects:

- It does not account for variation in the costs and priors associated with application of the recognizer, and
- it does not allow for analysis of performance in terms of discrimination and calibration.

NIST has addressed the first issue by defining the primary evaluation criterion C_{avg}, which can be interpreted as a mixture of the error-rates obtained when different priors are applied. It measures the ultimate practical decision-making ability of the technology and as such is a test of both discrimination and calibration of the systems under evaluation. Below we will criticize some language recognition evaluation practices, but we exclude C_{avg} from this criticism. In our opinion C_{avg} is a useful measure and indeed the whole object of the exercise described below is to better understand and to improve system performance as measured with C_{avg}.

The problems that we discuss below occur when evaluation criteria are chosen also to perform some form of calibration/discrimination analysis. Presumably in response to this need, several authors reporting on the series of NIST Language Recognition Evaluations have adopted the solution of *pooling language detection scores over multiple targets*[2] and then analyzing the pooled scores with the ROC analysis tools borrowed from speaker recognition. (Under ROC analysis, we also mean DET-curves as well as associated measures of goodness such as EER and 'min DCF'.)

Unfortunately, in our opinion, this pooled score analysis does not give the same useful insights into system performance as it does in speaker recognition. Below we mention the problems we see with pooling and then present the alternative evaluation analysis that we used to guide our development decisions for this LRE.

10.1. The problems with pooling

All ROC-curves are by definition pooled over multiple trials, but the question is: What do you pool and what don't you pool? In speaker recognition, where ROC analysis works well, the pooling is across different speakers and across different sessions of those speakers. Why can we not do the same for language and pool across all sessions of all languages?

Pooling is useful for ROC analysis in situations where it is reasonable to set a *single* decision threshold for all trials that are included in the pooling. The ROC curve (and everything derived from it) exercises a single score threshold to make decisions. By sweeping this threshold it does two things simultaneously:

- it accounts for variable relative weighting (by prior and cost) of P_{miss} and P_{fa} and
- it accounts for any monotonic rising miscalibration of the scores.

ROC analysis assumes that for each relative weighting of the error-rates, there is a single well-performing threshold for all of the trials that are pooled.

This assumption is broken by pooling trials over different languages, because of the way miscalibration occurs in a battery of language recognizers. With the current LRE detection evaluation setup, ROC analysis is done effectively on language posteriors, while miscalibration already occurs (*before* applying Bayes' rule) with the language likelihoods. When the likelihood for one language is overoptimistically large, the posterior for that language (which is what is thresholded by ROC) also becomes overoptimistically large but the posteriors for all other languages become pessimistic. It makes no sense to try to adjust a single threshold for all posteriors, because they compete and the optimal thresholds move in opposite directions in response to miscalibrations of the likelihoods.

Would it help to use a different threshold for each target? Yes it would, but then one would need to do a separate ROC analysis for every target and one would be swamped by DET-curves. This separate ROC solution is furthermore not theoretically very satisfying. If the different posteriors are effectively individually calibrated one ends up with a posterior of which the elements don't sum to one. This illegal posterior suggests that a calibration solution which produces a correctly normalized posterior may be preferable, from both a theoretical and practical viewpoint.

In summary, pooled ROC analysis of the posterior is problematic and this is not even solved in a satisfactory way by 'unpooling'. For further discussion see [11, 12].

[2] Note, C_{avg} pools error-rates, not scores and is therefore not subject to this criticism.

10.2. Our solution

The solution which we used to guide our development is based on [11]. We model (and correct for) calibration of the language *likelihoods*, rather than the language posteriors. By calibrating before the non-linearity of Bayes' rule, we can manage to do effective calibration analysis with a simple linear method.

Since we are optimizing for C_{avg} in this LRE, our development testing evaluation strategy was built around C_{avg}:

1. When we are busy with basic recognizer development (i.e. the frontends) we want to judge the *discrimination* rather than the calibration of our algorithms, In this case, we prefer not to use the calibration-sensitive C_{avg} as is. Our solution is to discount the effect of calibration by letting the *evaluator* calibrate every system. That is, the evaluator optimizes calibration on the evaluation data[3] and then reports the value of C_{avg} obtained with this calibration. We denote this measure by C^*_{avg}.

   The evaluator's calibration transformation involves *only scaling and translation* of the log-likelihood score-vectors, so that it does not alter the ability of the scores to discriminate between classes. In particular, the calibration transformation is *invertible*, so it does not alter the information content of the scores.

   In summary C^*_{avg} measures discrimination, not calibration. It is therefore similar in spirit to the EER (equal-error-rate) and 'min DCF' of speaker recognition, but it avoids the above-mentioned problems of score pooling.

2. When we design the final stage of the backend, we are more concerned with *calibration* than with discrimination. We judge calibration by observing the difference between C_{avg} and C^*_{avg}. When they are close, we judge calibration to be good.

3. Once all frontends and backends were made respectively as discriminative and as well-calibrated as we could get them, we made our final judgments about what to submit on C_{avg}.

11. Acknowledgments

This work was partly supported by European projects MOBIO (FP7-214324) and AMIDA (FP6-033812), by Grant Agency of Czech Republic project No. 102/08/0707, by Czech Ministry of Education project No. MSM0021630528 and by US Air Force European Office of Aerospace Research & Development (EOARD) Grant No. 083066.

12. References

[1] N. Brummer et al., "Discriminative Acoustic Language Recognition via Channel-Compensated GMM Statistics", submitted to Interspeech 2009. 3

[2] P. Matejka, L. Burget, P. Schwarz, and J. Cernocky: Brno University of Technology System for NIST 2005 Language Recognition Evaluation, in Proc. Odyssey 2006, San Juan, Puerto Rico, USA, June 2006. 3, 4

[3] P.A. Torres-Carrasquillo, E. Singer, M.A. Kohler, R.J. Greene, D.A. Reynolds, and J.R. Deller Jr., "Approaches to language identification using Gaussian mixture models and shifted delta cepstral features," in *Proc. International Conferences on Spoken Language Processing (ICSLP)*, Sept. 2002, pp. 89–92. 3

[3] Our MATLAB code to perform this optimization is freely available at http://niko.brummer.googlepages.com/focalmulticlass.

[4] P. Kenny, G. Boulianne, P. Ouellet, and P. Dumouchel, "Joint factor analysis versus eigenchannels in speaker recognition", IEEE Transactions on Audio, Speech and Language Processing 15 (4), pp. 1435-1447, May 2007. 3

[5] P. Kenny, N. Dehak, P. Ouellet, V. Gupta, and P. Dumouchel, "Development of the Primary CRIM System for the NIST 2008 Speaker Recognition Evaluation", in Proc. Interspeech 2008, Brisbane, Australia, Sept 2008. 3

[6] O. Glembek, L. Burget, N. Dehak, N. Brummer and P. Kenny, "Comparison of Scoring Methods used in Speaker Recognition with Joint Factor Analysis" in Proc. ICASSP 2009, Taipei, Taiwan, April 2009. 3

[7] P. Matějka, L. Burget, P. Schwarz, and J. Černocký, "Brno University of Technology system for NIST 2005 Language recognition evaluation," in *IEEE Odyssey: The Speaker and Language Recognition Workshop*, San Juan, Puerto Rico, June 2006, pp. 57–64. 3

[8] D.A. Reynolds, T.F. Quatieri, and R.B. Dunn, "Speaker verification using adapted Gaussian mixture models," *Digital Signal Processing*, vol. 10, no. 1–3, pp. 19–41, 2000. 3

[9] C.M. Bishop, *Pattern Recognition and Machine Learning*. Springer, 2007. 2

[10] J. Nocedal, and S.J. Wright, *Numerical Optimization*. Springer, 2006. 2

[11] N. Brümmer and D. van Leeuwen, "On calibration of language recognition scores", in Proc. IEEE Odyssey 2006: The Speaker and Language Recognition Workshop, San Juan, June 2006. 4, 5

[12] D. van Leeuwen and K.P. Truong, "An open-set detection evaluation methodology applied to language and emotion recognition". in Proc. Interspeech, pages 338341, Antwerp, August 2007. 4

[13] P. Schwarz, P. Matějka, and J. Černocký, "Towards lower error rates in phoneme recognition," in *Proc. International Conference on Text, Speech and Dialogue*, Brno, Czech Republic, Sept. 2004, pp. 465–472. 3

[14] T. Thomas, V. Wan, L. Burget, M. Karafit, J. Dines, J. Vepa, G. Garau and M. Lincoln: "The AMI System for the Transcription of Speech in Meetings", In: Proc. IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP 2007), Honolulu, 2007, pp. 357-360. 3

[15] J.L. Gauvain, A. Messaoudi and H. Schwenk, "Language Recognition using Phone Lattices," in *Proc. International Conferences on Spoken Language Processing (ICSLP)*, Sept. 2004, pp.1283–1286. 3

[16] J. Navratil: "Spoken language recognition-a step toward multilinguality in speech processing", in IEEE Trans. on Speech and Audio Processing, Vol. 9, No. 6, pp. 678-685 ISSN: 1063-6676, September 2001. 4

[17] J. Navratil: "Recent advances in phonotactic language recognition using binary-decision trees," in *Proc. International Conferences on Spoken Language Processing (ICSLP)*, Pittsburgh, PA, October 2006 4

[18] D. Povey, "Discriminative Training for Large Vocabulary Speech Recognition," Ph.D. thesis, Cambridge University, Jul. 2004. 3

[19] L. Welling, S. Kanthak and H. Ney, "Improved methods for vocal tract normalization", in Proc. ICASSP 1999. 3

[20] F. Castaldo, E. Dalmasso, P. Laface, D. Colibro and C. Vair: Language identification using acoustic models and speaker compensated cepstral-time matrices, Proc. ICASSP 2007. 4

[21] P. Schwarz, P. Matějka, and J. Černocký, "Hierarchical structures of neural networks for phoneme recognition," in *Proc. ICASSP*, Toulouse, France, May 2006, pp. 325-328. 3

[22] V. Hubeika, L. Burget, P. Matjka and P. Schwarz, " Discriminative Training and Channel Compensation for Acoustic Language Recognition", in Proc. Interspeech 2008. 3, 4

[23] V. Hubeika, L. Burget, P. Matejka and J. Cernocky: "Channel Compensation for Speaker Recognition, poster at MLMI 2007", Brno, June 2007. 4

[24] B. Zhang, S. Matsoukas, R. Schwartz: "Recent progress on the discriminative region-dependent transform for speech feature extraction", in Proceedings of INTERSPEECH, Pittsburgh, PA, September, 2006 3

[25] D. Povey: "fMPE: discriminatively trained features for speech recognition," in Proceedings of ICASSP, Philadelphia, PA, Mar. 2005, IEEE. 3

[26] W.M. Campbell, F. Richardson, and D.A. Reynolds: "Language Recognition with Word Lattices and Support Vector Machines", in Proc. ICASSP 2007. 4

Table 1: *Training data in hours for each language and source.*

Language	CTS		VOA	
	#files	#hours	#files	#hours
alba	0	0	104	3.4
amha	0	0	1724	77.7
arab	4085	201.8	0	0
azer	0	0	510	29.3
bang	213	5.2	3871	83.4
bosn	0	0	268	7.0
burm	0	0	3365	81.6
cant	482	6.9	34	2.1
creo	0	0	425	14.8
croa	0	0	150	5.3
czec	241	0.3	0	0
dari	0	0	2410	78.8
engi	714	2.2	0	0
engl	10560	290.9	3963	132.5
fars	656	22.6	0	0
fren	403	21.8	3679	88.7
geor	0	0	100	4.7
germ	685	23.1	0	0
gree	0	0	851	16.6
haus	0	0	2599	74.4
hind	755	26.0	358	15.7
hung	287	0.4	0	0
chin	1226	29.9	0	0
indo	267	0.4	226	3.0
ital	294	1.3	0	0
japa	718	23.1	0	0
khme	0	0	1297	53.0
knkr	0	0	1307	66.7
kore	691	21.3	342	16.3
mace	0	0	344	15.1
mand	1321	64.8	1049	35.7
ndeb	0	0	945	64.4
orom	0	0	399	15.1
pash	0	0	6317	102.3
pers	0	0	1673	70.6
poli	284	0.4	0	0
port	294	0.5	1069	48.7
russ	643	8.4	3071	82.2
serb	0	0	175	2.9
shon	0	0	553	58.6
soma	0	0	1909	70.9
span	1001	47.5	1623	67.6
swah	194	0.3	1965	70.9
swed	290	0.5	0	0
taga	24	0.6	0	0
tami	623	19.6	0	0
thai	209	6.6	0	0
tibe	0	0	349	2.0
tigr	0	0	395	24.6
turk	0	0	262	9.8
ukra	0	0	105	3.0
urdu	24	1.4	1242	67.2
uzbe	0	0	241	3.5
viet	743	25.7	113	8.9
SUM	27927	853.7	51382	1696.8