Factor Analysis and SVM for Language Recognition

Florian Verdet[1,2], *Driss Matrouf*[1]
Jean-François Bonastre[1], *Jean Hennebert*[2]

[1]Université d'Avignon et des Pays du Vaucluse, Laboratoire Informatique d'Avignon, France
[2]Département d'Informatique, Université de Fribourg, Fribourg, Switzerland

florian.verdet@univ-avignon.fr, driss.matrouf@univ-avignon.fr,
jean-francois.bonastre@univ-avignon.fr, jean.hennebert@unifr.ch

Abstract

Statistic classifiers operate on features that generally include both, useful and useless information. These two types of information are difficult to separate in feature domain. Recently, a new paradigm based on Factor Analysis (FA) proposed a model decomposition into useful and useless components. This method has successfully been applied to speaker recognition tasks. In this paper, we study the use of FA for language recognition. We propose a classification method based on SDC features and Gaussian Mixture Models (GMM). We present well performing systems using Factor Analysis and FA-based Support Vector Machine (SVM) classifiers. Experiments are conducted using NIST LRE 2005's primary condition. The relative equal error rate reduction obtained by the best factor analysis configuration with respect to baseline GMM-UBM system is over 60 %, corresponding to an EER of 6.59 %.

Index Terms: language recognition, UBM, Factor Analysis, SVM.

1. Introduction

The focus of this paper is language recognition, which consists in processing a speech signal to detect which language the speaker is talking in.

It is obvious that the data we observe includes not only useful information, but also information that doesn't help in the task of language recognition. The unwanted information covers speaker specificities including vocal tract configuration, current emotion or health status. It covers as well recording conditions with background noise, microphone setup, transmission channel and speech encoding. We propose here to qualify this perturbing information as *session* dependent. The data we observe is then composed of useful information, that is the information that depends on the language, and useless or even perturbing information, that is the information that depends on the session.

The feature extraction and modeling strategy (e.g. with GMMs) attempt to focus on the useful information and to discard the language independent perturbing information. However, usual feature extraction approaches can only partially discard perturbing information related to the recording setup and transmission channel. Furthermore, a lot of the speaker dependent specificities are kept in the features. The overall strategy proposed in this work is to keep track of this session variability. This helps distinguishing the language dependent information.

For this, we need a few sessions for every language (typically, each utterance recording can be seen as a different session). In order to detach language dependent information from session variability, we consider the language part being the information that is common to all sessions (utterances) and the residuals being the session variability.

In testing stage, where we're left alone with only one session, the variability contained in the data is estimated and removed based on the variability seen in the training data. What remains should hopefully emphasize the useful part of the information and thus, the classification should be more precise.

These principles can actually be implemented using the *Factor Analysis* (FA) approach. FA has triggered significant advances in speaker verification as described in [2, 1]. In this paper we describe how to implement factor analysis for language recognition and we evaluate its behavior and benefits on language recognition in a realistic task. The context is sensibly different from speaker recognition. Since each class has far more training data, we will probe bigger models.We have less classes (only some languages instead of a lot of speakers), but there are a lot of different sessions for each language. As we have, in our setting, a different speaker from session to session, our experiments also analyze if this big speaker variability may still be caught at the same time as the finer (e.g. channel) variabilities.

GMMs as well as FA are statistical modeling techniques, which try to mold at best the clusters of points in multidimensional space. Another kind of techniques is discriminative classification as artificial neural networks or support vector machines (SVM), which seek to trace the decision boundaries between the clusters, thus working on the difficult part of space where most of the confusions occur. In speaker verification, SVM systems slightly outperform FA. This paper will also try to verify if this conclusion holds for language recognition. The models obtained with the FA approach can directly be used in a SVM classifier. This association allows to benefit from the FA decomposition power and the SVM classification power.

All reported experiments are conducted using the free software framework MISTRAL [4], which uses the ALIZE library [5]. The evaluation protocols are the ones of NIST Language Recognition Evaluation 2005.

Section 2 gives a description of the different systems studied in this paper: UBM-based GMMs in 2.1, UBM-based FA in 2.2 and FA-based SVMs in 2.3. Section 3 introduces the parametric setup and the databases used for the evaluation. The results and their discussions are presented in Section 4.

2. System descriptions

2.1. GMM system using a UBM

To keep our models general enough to cover also feature vectors which haven't been seen during training, the models are based on a *Universal Background Model* (UBM, also known as world model). The parameters of the UBM are estimated taking as much and as different data as possible from a large set of languages. The model for each language is then obtained through a *Maximum A-Posteriori* (MAP) *adaptation* of this very general UBM towards the training data. For the mean values (μ) of the Gaussians, the MAP adaptation can be expressed as a weighted sum of the UBM mean and the mean of one language's training data (factor α being the importance of the training data mean):

$$\mu_{language} = (1-\alpha) * \mu_{ubm} + \alpha * \mu_{data} \tag{1}$$

This keeps both, a certain generality coming from the UBM side and a good fitting of the training data of the language.

2.2. GMM-UBM with Factor Analysis

For simplicity and because it has proven to work well, we operate the factor analysis solely on the means of the Gaussian mixtures [1]. If we concatenate all the means of one model, we obtain one big *mean super-vector* (SV). The basic factor analysis (FA) formula can be stated as:

$$m_{observed} = m_{ubm} + Dy_{language} + Ux_{session} \tag{2}$$

where m are mean supervectors, y is the part which is specific to the language, weighted by D, and Ux is the session variability, which is included in the observed data but which we don't want to include in the language model. The Factor Analysis model assumes that the session variability is located in low-dimensional subspace. This subspace is generated by the vector columns of the U matrix. x are the session factors in this subspace.

2.2.1. Estimating eigensession compensation matrix

The matrix U, here called *eigensession matrix*, is common to all languages. It is iteratively estimated using expectation maximization (EM) algorithm with maximum likelihood (ML) optimization criterion. Each step, the different $x_{session}$ and $y_{language}$ are estimated, then U is estimated globally, based on these x and y. Since x and y also depend on U, the process is iterated. The step by step algorithm is described in [1].

2.2.2. Training language models

The model for each language which is stored at training stage is the $m+Dy$ part of factor analysis formula (2). It is, very similar to MAP adaptation, a weighted combination of the UBM mean m and the mean supervector y estimated using all training data corresponding to the target language. D being the weighting factor. It can be seen as MAP adaptation operating on session-compensated (Ux part removed) information.

For a given language, the per-session x are estimated. Using the stored U matrix, the Ux part is subtracted from the data's mean supervector and y gets estimated. Finally, the remaining $m + Dy$ part is stored as the language's model means (see [1] for details). To obtain full language model parameters, mixture weights and covariances are kept unchanged from UBM.

2.2.3. Testing using compensated models

We assume that the observed feature vectors contain session perturbation (speaker and channel). In testing stage, different strategies may be applied to cope with this:

Standard feature normalization The most common strategy consists in normalizing each feature vector by removing the session effect, which is the Ux component weighted by the feature vector's posterior probability against hypothesized language's model. x is estimated using statistics of the testing utterance as detailed in [1]. Once the feature vectors are normalized, they are tested against the model of the hypothesized language. We may sketch this by following:

$$feat - Ux_{utterance} \quad | \quad m + Dy_{language}$$

So, we try to remove the impurity (normalizing by Ux component) to match them up with the clean models (which don't include session variability Ux) issued from training stage.

Model compensation Instead of removing session variability from the features, we may likewise compensate this variability on the models themselves:

$$feat \quad | \quad m + Dy_{language} + Ux_{utterance}$$

We simply add the Ux component to the model (we're still in mean supervector context). So that un-normalized test data is tested against a modified model. This is not the same as saving the full model (including Ux) at training stage, since here, x is estimated on current test utterance only.

Client based model compensation This idea can be spun further: For estimating x, the factor analysis formula (2) is resolved to x, which implies subtracting $m + Dy$ from current utterance's mean. Since usually y is initialized to zero (see [1]), the Dy term disappears and x catches, mapped to SV-space and thus constrained by U, as much of the difference between utterance's and UBM's means as it can. The formula, which x estimation is based on is therefore:

$$m_{observed} = m_{ubm} + Ux$$

This is likely not the best solution. Thus, we initialize y by extracting it from the stored model of the hypothesized language (remember, it was stored as the $m + Dy$ part). We know m, the UBM mean supervector, and the weighting D. So y is easy to reconstruct (we call it $y_{language}$ because it is fetched from the language model). Now, we may rewrite the factor analysis formula, which x estimation is based on:

$$m_{observed} = m_{ubm} + Dy_{language} + Ux$$

Since y now depends on the language, x has to be estimated separately for each utterance-language combination test.

While x being estimated this way, the stored language models are compensated as described for model compensation strategy before being used for testing the utterance against. In all presented experimental results, this strategy has been adopted.

2.3. Support vector machines

Instead of having models working in feature vector space, support vector machines (SVM) work directly with the points representing mean super-vectors (SV) in multidimensional space. For this, we not only have one SV representing the language of interest, but it stays in contrast with a set of impostor SVs (*blacklist*) and which represents all other possible languages.

SVM training tries to find some kind of hyperplane in this space which best separates the target from the impostor SVs. In practice, there is a large variety in kinds of hyperplanes, called kernels. We use the most simple one, which is the *linear kernel*. Here again, similarly as for GMM-UBM, the decision boundary must not over-fit training data to remain robust to the natural variability of each language's data. The SVM principle for testing stage is to transform the input utterance to a temporary model using MAP or FA strategy and then stacking the mean values of this model's Gaussians to produce the SV which will be checked against the decision boundary.

Different setups of target SVs and blacklist compositions can be imagined. Some are described in the next paragraphs.

2.3.1. Impostors blacklist

In a first approach, the mean SVs of the FA language models can be used as input for SVM training. Operating on L languages, we will have one target SV and a blacklist composed of the $L-1$ SVs representing the other languages.

Instead of forming the blacklist with only a few mean SVs coming from the other models, we will try to use more SVs. We may generate a lot of impostor SVs by transforming each of the (impostor) training utterances separately. The SVs are obtained applying the same process as in testing stage.

Ideally, the data in the blacklist should even be distinct from the one used for training the other models. In this case, it would be straightforward to have a lot of impostor SVs.

2.3.2. Target vectors

This development can also be applied to the target SV, so we may use not only one target SV, but multiple target SVs. These are obtained, this time not from the underlying (GMM or FA) language models, but from the language's training utterances.

2.4. Scoring

Scores are normalized separately for each test utterance among all languages. This is done by dividing each score (usually the likelihood of the test utterance being of a given language) by the sum over the scores obtained against the other language models.

System performance can be enhanced by powering each score with a constant K. Matějka et al. describe in [3] that this procedure attempts to introduce some correction to the assumption of the frames being independent to each other.

$$\widehat{score}_l(utterance) = \frac{score_l(utterance)^K}{\sum\limits_{i \neq l \in L} score_i(utterance)^K}$$

l being the hypothesized language and i being of the other languages.

This normalization is also applied to the scores of the SVM systems, even if the effect is far less crucial.

2.5. Evaluation

System performance is measured at equal error rate operating point. It is the detection system choosing the decision threshold in such a way that the rate of false acceptances (accepted impostor utterances) is equal to the rate of false rejects (client utterances not recognized as being of the true language). This is denoted as *Equal Error Rate* (EER) and is usually expressed as percentage.

3. Experimental setup

The framework used for all experiments is principally the free software MISTRAL [4, 5]. For cepstral feature extraction, SPro4 [6] has been used.

All experiments are run in the context of a 7-language recognition to be comparable with NIST's Language Recognition Evaluation (LRE) 2005 [11], primary condition. The seven languages are: English, Hindi, Japanese, Korean, Mandarin, Spanish and Tamil.

3.1. Parametric setup

In our experiments, we use *Shifted Delta Cepstra* (SDC) parameters in the configuration 7-1-3-7 (in concordance with other researches in this domain [7, 8, 9, 3]). This means we're having 6 cepstral MEL-scale coefficients plus energy (cepstra and energy are kept in the parametric vector) and seven delta blocks stacked, each block calculated on frames $t-1$ and $t+1$ with a t shifted by 3 each time. This yields feature vectors of size 56.

Speech detection is conducted on all utterances to spot speech and non-speech parts. It is based on the energy and is done by a three-Gaussian classifier taking half of the central Gaussian as speech. A slight smoothing of the speech segmentation is then performed to clean up far too short segments.

All features are then normalized in such a way that the features containing speech of one utterance have an average of 0 and a variance of 1.

Some experiments (not presented here) have been conducted using linear scale filters and also standard MFCC or LFCC parameters with energy, their deltas and accelerations. But the chosen MEL-scale based SDC parameters have proven to carry most useful information.

3.2. Data parts

3.2.1. Training data

For training, all three parts (train, devtest and evltest) of the CallFriend [10] corpus are used. Each of these three parts of the corpus contains 20 complete two-ended, half-hour conversations per language. The corpora of the named languages are used, including both available dialects for English, Mandarin and Spanish (having thus 40 conversations). 42.2 % of the data being detected as speech, we have about 20 (resp. 40) hours of speech for each language.

3.2.2. Testing data

As announced, tests are conducted on NIST-LRE 2005 data [11]. This evaluation set contains 10986 utterances containing 3, 10 and 30 seconds of speech each. The primary condition aggregates just utterances of the seven languages (closed-set condition) with a total of 10734 utterances. We focus mainly on the 30 second ones, which comes down to 3578 files giving that many target trials and thus 21468 impostor trials.

3.3. Universal Background Model

The UBM is first roughly initialized doing 2 EM/ML-iterations selecting randomly one tenth of all training feature vectors. Then it is refined during 20 iterations with all training data.

4. Results

While development has been done on GMMs with 256 Gaussians, results are also presented featuring full systems using 2048 Gaussians. All results are for 30 second segments, NIST LRE 05's closed-set primary condition.

4.1. MAP adapted GMM-UBM system

The GMM-UBM language models are obtained with 10 iterations of MAP adaptation, where only the mean values are changed (neither Gaussians' weights nor variances are adapted).

While seeing the GMM-UBM system as baseline, it obtains 19.51 % EER using 256 Gaussians and 17.05 % EER with 2048, which represents about 12 % relative gain.

4.2. UBM-based factor analysis system

For the FA system, the eigensession matrix is obtained using the UBM and is set to have a *rank* of 40 (number of session factors). It is iteratively estimated during 20 iterations.

For training each language model, statistics over training data are collected and x and y are estimated. The model, being the $m + Dy$ part, gets then fashioned in one iteration.

This factor analysis system performs at 9.64 % EER using mixtures of 256 Gaussians. As Table 1 shows, it jumps to 6.59 % EER with 2048 Gaussians, which is a far bigger improvement (30 % relative) as observed for GMM-UBM systems. While the capacity of GMM systems seem to exhaust with about 512 Gaussians, FA systems begin to reveal their power by increasing model size. In the range of evaluated setups, reduction in error rate seems to be linear to the number of Gaussians added to the models. Observing this big performance impact of FA over the baseline GMM-UBM system validates the profit of FA for language recognition.

Table 1: *GMM-UBM and UBM-based FA systems with different model sizes (256 to 2048 Gaussians; ratings in % EER).*

system	256	512	1024	2048
GMM-UBM	19.51	18.36	17.69	17.05
FA	9.64	9.25	8.33	6.59

4.3. FA-based SVM systems

Presented are results using one target SV issued from the FA model or using multiple target SVs, where every training utterance of the target language is represented by an own SV.

These setups are combined with three different blacklist configurations:

Set A is composed of one SV for each of the non-target languages. Since we work in 7 language context, we have 6 SVs in this type of blacklist. Each SV is obtained by stacking the Gaussians' means of the FA language model.

Set B is composed of one SV for each training utterance of the non-target languages. Since the training corpus comprises all three CallFriend parts, these blacklists contain 960 or 1080 SVs, issued from utterances containing about 12 minutes of speech.

Set C is the same as set B augmented by all SVs of non-target language utterances of NIST LRE 2003 evaluation data containing utterances of 3, 10 and 30 seconds. Blacklists count thus between 2640 and 3240 SVs.

Exploring these combinations is motivated by the fact that estimating a separating hyperplane using only one point on the target side seems amendable and for the impostor side, using more SVs, we expect better generality (similar to a world model adducting robustness).

Table 2 shows the results of the 256 Gaussian SVM systems with different target SVs and blacklist combinations. All SVs are the mean supervectors obtained of FA models. One test has been conducted on 2048 Gaussians: For one target SV and blacklist set C, the EER is at 5.61 %.

The results primarily indicate that the blacklist should be composed of as different SVs as possible (set C). In addition, we obtain better results using multiple target SVs, given we're also having enough impostor SVs. We see that FA-based SVM systems outperform FA systems also in language recognition, but they are more difficult to tune and the profit may be lost.

Table 2: *SVM results for 256 Gaussians, in % EER.*

blacklist	one target SV	multiple target SVs
set A	8.16	8.74
set B	7.32	7.88
set C	7.12	6.93

5. Conclusions

In this paper, we investigated the use of factor analysis method for language recognition. We compared standard GMM-UBM approach, GMM-UBM combined with FA, and FA-based SVM setups. Experiments on NIST LRE 2005 demonstrated the efficiency of the proposed approach: The basic GMM-UBM approach with 2048 Gaussians gives an EER of 17 %. With UBM-based FA the EER is 6.59 % and on the FA-based SVM system, it is 5.61 %, which is a relative gain of 67% over bare GMM.

These results suggests that, even if it was mainly applied to speaker verification issue, factor analysis is a general approach of variability reduction that could be successfully applied to various pattern recognition tasks. Starting from this idea, we now plan to generalize the proposed approach by integrating it to other feature types.

For Factor Analysis, first experiments show that increasing the eigensession matrix' rank doesn't improve much, but with considerable computational effort, even bigger models should be investigated since we don't observe any stagnation yet.

While SVM based on FA works well, we should, as perspective, find some way to combine FA with the promising Maximum Mutual Information (MMI) training.

This work is supported by MOBIO, the europeen project FP7-2007-ICT-1.

6. References

[1] Matrouf, D., Scheffer, N., Fauve, B., Bonastre, J.-F., "A straightforward and efficient implementation of the factor analysis model for speaker verification", in: Interspeech 2007, 1242-1245, 2007.

[2] Kenny, P., Boulianne, G., Ouellet, P., Dumouchel, P., "Factor Analysis Simplified", in: Proc. of ICASSP '05., vol.1, pp. 637-640, March 18-23, 2005.

[3] Matějka P., Burget L., Schwarz P., Černocký J., "Brno University of Technology System for NIST 2005 Language Recognition Evaluation", in: Proc. of Odyssey 2006: The Speaker and Language Recognition Workshop, San Juan, PR, 57-64, 2006.

[4] The MISTRAL project, open source platform for biometrics authentification, http://mistral.univ-avignon.fr

[5] Bonastre, J.-F., Wils, F., Meignier, S., "ALIZE, a free toolkit for speaker recognition", in: Proc. of ICASSP '05, vol.1, pp. 737-740, March 18-23, 2005.

[6] SPro, a free speech signal processing toolkit, http://www.irisa.fr/metiss/guig/spro/

[7] Burget, L., Matejka, P., Cernocky, J., "Discriminative Training Techniques for Acoustic Language Identification", in: Proc. of ICASSP '06, vol.1, pp.I-I, 14-19 May 2006

[8] Campbell, W. M., Singer, E., Torres-Carrasquillo, P. A., Reynolds, D. A., Language Recognition with Support Vector Machines, in: Proc. of Odyssey: The Speaker and Language Recognition Workshop, Toledo, Spain, ISCA, pp. 4144, 31 May3 June 2004.

[9] Castaldo, F., Colibro, D., Dalmasso, E., Laface, P., Vair, C., "Compensation of Nuisance Factors for Speaker and Language Recognition", Audio, Speech, and Language Processing, IEEE Transactions on, vol.15, no.7, 1969-1978, Sept. 2007.

[10] CallFriend corpus, telephone speech of 15 different languages or dialects, http://www.ldc.upenn.edu/Catalog

[11] The 2005 NIST Language Recognition Evaluation, evaluation plan, http://www.itl.nist.gov/iad/mig/tests/lre/2005