IMPROVING SINGING LANGUAGE IDENTIFICATION THROUGH I-VECTOR EXTRACTION

Anna M. Kruspe

Fraunhofer IDMT, Ilmenau, Germany
Center for Language and Speech Processing, Johns Hopkins University, Baltimore, USA
kpe@idmt.fhg.de

ABSTRACT

Automatic language identification for singing is a topic that has not received much attention in the past years. Possible application scenarios include searching for musical pieces in a certain language, improvement of similarity search algorithms for music, and improvement of regional music classification and genre classification. It could also serve to mitigate the "glass ceiling" effect. Most existing approaches employ PPRLM processing (Parallel Phone Recognition followed by Language Modeling).
We present a new approach for singing language identification. PLP, MFCC, and SDC features are extracted from audio files and then passed through an i-vector extractor. This algorithm reduces the training data for each sample to a single 450-dimensional feature vector. We then train Neural Networks and Support Vector Machines on these feature vectors. Due to the reduced data, the training process is very fast. The results are comparable to the state of the art, reaching accuracies of 83% on a large speech corpus and 78% on acapella singing. In contrast to PPRLM approaches, our algorithm does not require phoneme-wise annotations and is easier to implement.

1. INTRODUCTION

Language Identification (LID) describes the task of automatically detecting the language spoken in an audio document. In speech recognition, LID has been a topic of interest for more than 30 years and has been extensively researched. In the Music Information Retrieval field, a similar language identification can be performed for singing (Singing Language Identification, SLID). There have so far only been a handful of publications on SLID despite a number of interesting application scenarios, such as:

Direct search of music in a certain language SLID can be useful for private users who are, for example, looking for music for a holiday video, or for music to help them learn a language. Commercial users could use this for advertisement videos.

Improvement of similarity search Similarity dimensions could include the sung language.

Improvement of regional classification As mentioned in [1], human subjects tend to rely on the language to determine the region of origin of a musical piece. This is not taken into account by current regional classification systems.

Improvement of genre classification Similar to regional classification, certain musical genres are closely connected to a single singing language. Considering the “glass ceiling” of approximately 80% for most classification tasks[2], new hybrid approaches are necessary to improve them. SLID could serve this purpose, too.

Only a few SLID systems have been developed so far. They mostly use the principle of Parallel Phone Recognition followed by Language Modeling (PPRLM). In this paper, we present an approach that does not require the extensive annotations used in PPRLM. Our approach employs three commonly used audio features with Multi-Layer Perceptrons (MLPs) and Support Vector Machines (SVMs) as backend classifiers. Using the i-vector extraction algorithm as a processing step in between, our approach is able to surpass the state of the art.
We will give a more detailed overview over the state of the art in section 2 before presenting the used datasets in section 3. We then describe our proposed system in section 4 and show experimental results in section 5. Finally, we draw conclusions in section 6 and make suggestions for further research in section 7.

2. STATE OF THE ART

2.1. Language identification for speech

Language identification has been extensively researched in the field of Automatic Speech Recognition since the 1980’s. A number of successful algorithms have been developed over the years. An overview over the fundamental techniques is given by Zissman in [3].
Fundamentally, four properties of languages can be used to discriminate between them:

Phonetics The unique sounds that are used in a given language.

Phonotactics The probabilities of certain phonemes and phoneme sequences.

Prosody The “melody” of the spoken language.

Vocabulary The possible words made up by the phonemes and the probabilities of certain combinations of words.

Even modern system mostly focus on phonetics and phonotactics as the distinguishing factors between languages. Vocabulary is sometimes exploited in the shape of language models.
Zissman mentions Parallel Phone Recognition followed by Language Modeling (PPRLM) as one of the basic techniques. It requires audio data, language annotations, and phoneme annotations for each utterance. In order to make use of vocabulary characteristics, full sentence annotations and word-to-phoneme dictionaries are also necessary.
Using the audio and phoneme data, acoustic models are trained. They describe the probabilities of certain sound and sound sequences occurring. This is done separately for each considered language. Similarly, language models are generated using the sentence annotations and the dictionary. These models describe the probabilities of certain words and phrases. Again, this is done for each language.

New audio examples are then run through all pairs of acoustic and language models, and the likelihoods produced by each model are retained. The highest acoustic likelihood, the highest language likelihood, or the highest combined likelihood are then considered to determine the language. This approach achieves up to 79% accuracy for ten languages [4].
Another approach uses the idea to train Gaussian Mixture Models for each language. This technique can be considered a "bag of frames" approach, i.e. the single data frames are considered to be statistically independent of each other. The generated GMMs then describe probability densities for certain characteristics of each language. Using these, the language of new audio examples can be easily determined.
GMM approaches used to perform worse than their PPRLM counterparts, but the development of new features has made the difference negligible [5]. They are in general easier to implement since only audio examples and their language annotations are required. Allen et al. [6] report results of up to 76.4% accuracy for ten languages. Different backend classifiers, such as Multi-Layer Perceptrons (MLPs) and Support Vector Machines (SVMs) [7] have also been used succesfully instead of GMMs.

2.2. Special challenges in singing

Singing presents a number of challenges for language identification when compared to pure speech. To mention a few examples:

Larger pitch fluctuations A singing voice varies its pitch to a much higher degree than a speaking voice. It often also has very different spectral properties.

Higher pronunciation variation Singers are often forced by the music to pronounce certain sounds and words differently than if they were speaking them.

Larger time variations In singing, sounds are often prolonged for a certain amount of time to fit them to the music. Conversely, they can also be shortened or left out completely.

Different vocabulary In musical lyrics, words and phrases often differ from normal conversation texts. Certain words and phrases have different probabilities (e.g. higher focus on emotional topics in singing).

Background music adds irrelevant data (for language identification) to the signal, which acts as an interfering factor to the algorithms. It therefore should be removed or suppressed prior to the language identification, e.g. by source separation algorithms.
In this paper, we only work with a-capella music to remove this difficulty.

So far, only a few approaches to perform language identification on singing have been proposed.
Schwenninger et al. [8] use MFCC features, but do not mention how they perform their actual model training. They test different pre-processing techniques, such as vocal/non-vocal segmentation, distortion reduction, and azimuth discrimination. None of these techniques seem to improve the over-all results. They achieve an accuracy of 68% on a-capella music for two languages (English and German).
The approach of Tsai and Wang [9] follows a traditional PPRLM flow. After vocal/non-vocal segmentation, they run their data through acoustic models using vector tokenization. One acoustic model for each language is used. The results are then processed by bigram language models, again for each language. The language model score is used for a maximum likelihood decision to determine the language. They achieve results of 70% accuracy for two languages (English and Mandarin).
Mehrabani and Hansen [10] also use a PPRLM system, with the difference that all combinations of acoustic and language models are tested. Their scores are combined by a classifier to determine the final language. This results in a score of 78% for three languages (English, Hindi, and Mandarin). Combining this technique with prosodic data improved the result even further.
Chandrasekhar et al.[11] try to determine the language for music videos using both audio and video features. They achieve accuracies of close to 50% for 25 languages. It is interesting to note that European languages seem to achieve much lower accuracies than Asian and Arabic ones. English, French, German, Spanish and Italian rank below 40%, while languages like Nepali, Arabic, and Pashto achieve accuracies above 60%.
Finally, we previously tested a different system based on Gaussian Mixture Models (GMMs) [12]. This approach does not require phoneme-wise annotations like the PPRLM approaches and is easier to implement. We achieved an accuracy of 68% on three languages (a-capella data).

3. DATASETS

In order to test our system on singing data, we used the data set previously presented in [12]. It consists of a-capella songs downloaded from *YouTube*[1]. The songs are performed by amateur singers in the languages English, German, and Spanish. We call it *YTAcap*.
For comparison, we also tested our algorithm on two well-known speech data sets: The *2003 NIST Language Recognition Evaluation (NIST2003LRE)* corpus [13] and the *OGI Multi-language Telephone Speech Corpus (OGIMultilang)*[14], using only the three previously mentioned languages.
An overview over the amount of data across the three corpora is given in table 1.

Table 1: *Amounts of data in the three used data sets: Sum duration on top, number of utterances in italics.*

hh:mm:ss ***#Utterances***	**NIST2003LRE**	**OGIMultilang**	**YTAcap**
English	00:59:08 *240*	05:13:17 *1912*	08:04:25 *1975*
German	00:59:35 *240*	02:52:27 *1059*	04:18:57 *1052*
Spanish	00:59:44 *240*	03:05:45 *1151*	07:21:55 *1810*

[1]`http://www.youtube.com/`

4. PROPOSED SYSTEM

Figure 1 shows a rough overview over our classification system. In the following, we will describe the selected features, the i-vector extraction algorithm, and the selected training backends in more detail.

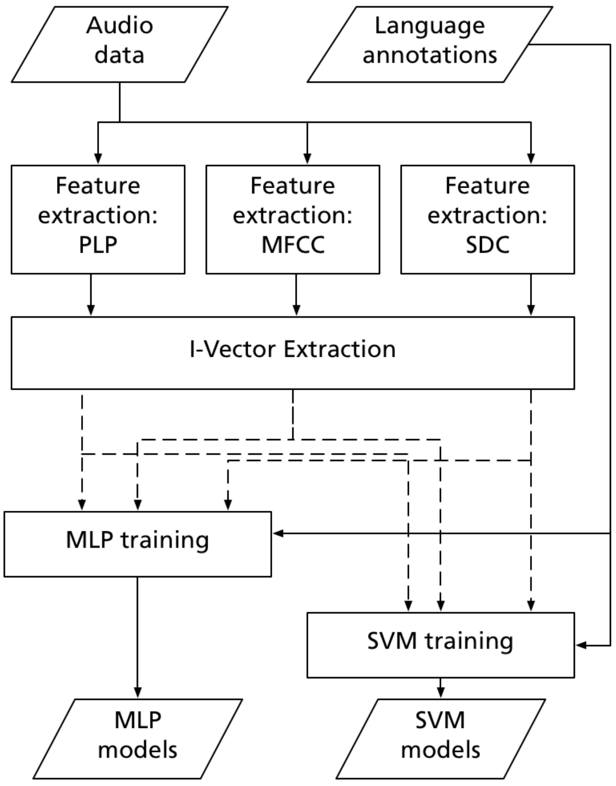

Figure 1: *Overview of the steps of our classification system.*

4.1. Features

We extracted a set of features for each audio file. Table 2 shows an overview over the various configurations used in training.

Perceptive Linear Predictive features (PLPs) PLP features, first introduced in [15], are among the most frequently used features in speech processing. They are based on the idea to use knowledge about human perception to emphasize important speech information in spectra while minimizing the differences between speakers. We use a model order of 13 in two experiments and one of 32 in another. Deltas and double deltas between frames are also calculated. We test PLPs with and without RASTA pre-processing [16].

Mel-Frequency Cepstral Coefficients (MFCCs) Just like PLPs, MFCCs are frequently used in all disciplines of automatic speech recognition [3]. We kept 20 cepstral coefficients for model training. Additionally, we calculated deltas and double deltas.

Shifted Delta Cepstrum (SDCs) Shifted Delta Cepstrum features were first described in [17] and have since been successfully used for speaker verification and language identification tasks on pure speech data [18] [7] [6]. They are calculated on MFCC vectors and take their temporal evolution into account. Their configuration is described by the four parameter $N-d-P-k$, where N is the number of cepstral coefficients for each frame, d is the time context (in frames) for the delta calculation, k is the number of delta blocks to use, and P is the shift between consecutive blocks. The delta cepstrals are then calculated as:

$$\Delta c(t) = c(t+iP+d) + c(t+iP-d), 0 <= i <= k \quad (1)$$

with $c \in [0, N-1]$ as the previously extracted cepstral coefficients. The resulting k delta cepstrals for each frame are concatenated to form a single SDC vector of the length kN. We used the common parameter combination $N = 7, d = 1, P = 3, k = 7$.

4.2. I-Vector extraction

I-Vector (identity vector) extraction was first introduced in [19] and has since become a state-of-the-art technique for various speech processing tasks, such as speaker verification, speaker recognition, and language identification [20]. To our knowledge, it has not been used for any Music Information Retrieval tasks before.
The main idea behind i-vectors is that all training utterances contain some common trends, which effectively add irrelevance to the data in respect to training. Using i-vector extraction, this irrelevance can be filtered out, while only the unique parts of the data relevant to the task at hand remain. The dimensionality of the training data is massively reduced, which also makes the training less computationally expensive. As a side effect, all feature matrices are transformed to i-vectors of equal length, eliminating problems that are caused by varying utterance lengths.
Mathematically, this assumption can be expressed as:

$$M(u) = m + Tw \quad (2)$$

In this equation, $M(u)$ is the GMM supervector for utterance u. The supervector approach was first presented in [21] and has since been successfully applied to a number of speech recognition problems. A music example can be found in [22]. m represents the language- and channel-independent component of u and is estimated using a Universal Background Model (UBM). T is a low-rank matrix modeling the relevant language- and channel-related variability, the so-called Total Variability Matrix. Finally, w is a normally distributed latent variable vector: The i-vector for utterance u.

Step 1: UBM training A Universal Background Model (UBM) is trained using Gaussian Mixture Models (GMMs) from all utterances. This UBM models the characteristics that are common to all of them.

Step 2: Statistics extraction 0th and 1st order Baum-Welch statistics are calculated for each of the utterances from the UBM according to:

$$N_c(u) = \sum_{t=1}^{L} P(c|y_t, \Omega) \quad (3)$$

$$\widetilde{F_c}(u) = \sum_{t=1}^{L} P(c|y_t, \Omega)(y_t - m_c) \quad (4)$$

Table 2: *Feature configurations used in training.*

Name	Description	Dimensions
PLP	PLP with RASTA processing, model order 13 with deltas and double deltas	39
PLP36	PLP with RASTA processing, model order 36 with deltas and double deltas	96
PLP_NORASTA	PLP without RASTA processing, model order 13 deltas and double deltas	39
MFCC	MFCC, 20 coefficients	20
MFCCDELTA	MFCC, 20 coefficients, deltas and double deltas	60
SDC	SDC with configuration $7-1-3-7$	91
MFCCDELTASDC	MFCCDELTA+SDC	117
COMB	PLP_NORASTA+MFCCDELTA	99

where $u = y_1, y_2, ..., y_L$ denotes an utterance with L frames, $c = 1, ..., C$ denotes the index of the Gaussian component, Ω denotes the UBM, m_c is the mean of the UBM mixture component c, and $P(c|y_t, \Omega)$ denotes the posterior probability that the frame y_t was generated by mixture component c. As the equation shows, the 1st order statistics are centered around the mean of each mixture component.

Step 3: T matrix training Using the Baum-Welch statistics for all utterances, the Total Variability Matrix T is now trained iteratively according to:

$$w = (I + T^t\Sigma^{-1}N(u)T)^{-1}T^t\Sigma^{-1}\widetilde{F}(u) \tag{5}$$

using Expectation Maximization.

Step 4: Actual i-vector extraction Finally, an i-vector w can be extracted for each utterance using equation 5 again. This can also be done for unseen utterances, using a previously trained T.

4.3. Classification backend

For classification, we tested Multi-Layer Perceptrons (MLPs) and Support Vector Machines (SVMs).
The MLPs were fixed at three layers, with the middle layer having a dimension of 256. They were implemented using Quicknet [23]. Additional layers did not seem to improve the result. A larger middle layer only improved it slightly.
The SVM parameters were determined using a grid-search. In the full-feature experiments, we additionally employed a previous feature selection using the "Inertia Ratio Maximization using Feature Space Processing" (IRMFSP) [24]. For both IRMFSP and SVMs, we used our own C++ implementation.

5. EXPERIMENTAL RESULTS

As described above, we performed experiments using both MLP and SVM classifiers on all three data sets (NIST, OGI, and YTA-cap). For each of those classifiers and data sets, we test all of the combinations of features listed in table 2 directly and with i-vector processing. All results were obtained using five-fold cross validation.

5.1. MLP results

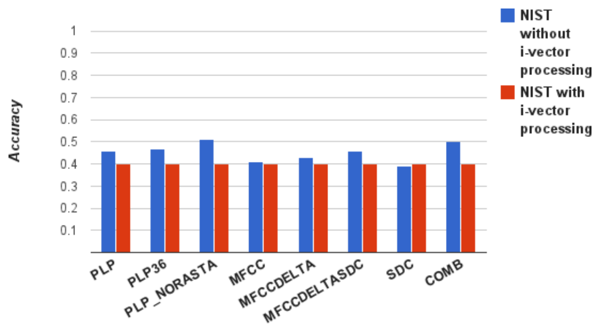

Figure 2: *Results for all feature combinations on the NIST2003LRE database, using MLP classifiers.*

NIST2003LRE As shown in figure 2, our MLP did not produce good results on the *NIST2003LRE* database for any of the feature combinations. *NIST2003LRE* is the smallest of the data sets by a large margin. Since we use a relatively high dimensional model, this is probably a case of overtraining. The i-vector processing step reduces the training data even further, thus aggravating the problem.

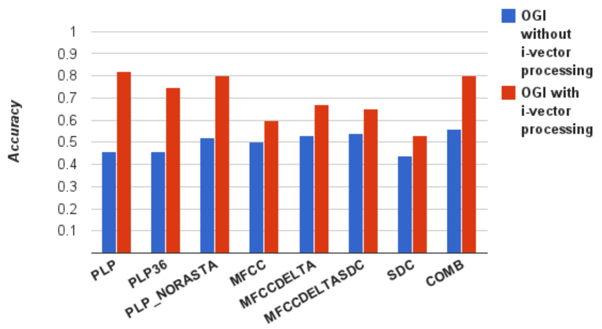

Figure 3: *Results for all feature combinations on the OGIMultilang database, using MLP classifiers.*

OGIMultilang The *OGIMultilang* data set contains roughly 4 times as much data as the *NIST2003LRE* set. With enough data, training an MLP classifier works a lot better. Without i-vector processing, we still only reach about 52% accuracy. I-Vector extraction improves the system massively. The best feature configurations are plain PLP (82%), PLP_NORASTA (80%) and COMB (80%).

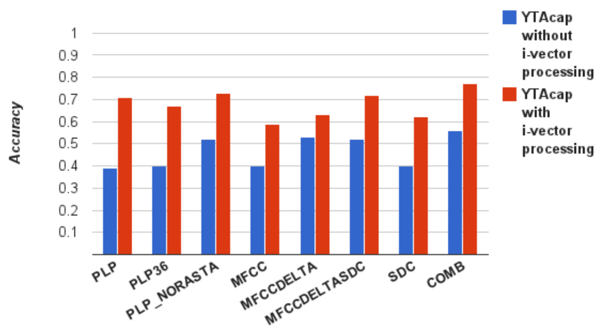

Figure 4: *Results for all feature combinations on the YTAcap database, using MLP classifiers.*

YTAcap In section 2.2, we described some factors that increase the difficulty for language identification on acapella data versus spoken data. As expected, the results on the *YTAcap* data set are somewhat worse than those on *OGIMultilang*, even though they contain a similar amount of data. The best result without i-vector extraction is still obtained using the COMB feature configuration at 56%. Similar to the *OGIMultilang* experiment, i-vector extraction yields a large improvement. COMB remains the best configuration, now at 77%.

5.2. SVM results

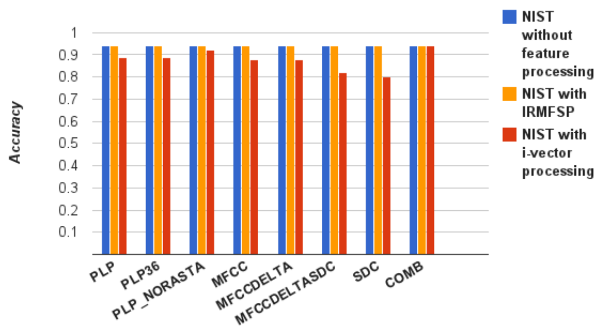

Figure 5: *Results for all feature combinations on the NIST2003LRE database, using SVM classifiers.*

NIST2003LRE In contrast to the MLP experiment, SVMs produced very good results on the *NIST2003LRE* data set for all of the features. They seem to be able to discriminate almost perfectly for this small, clean data set. We believe 94% might be an upper bound for the classification here, which might be caused by annotation errors or ambiguous data.

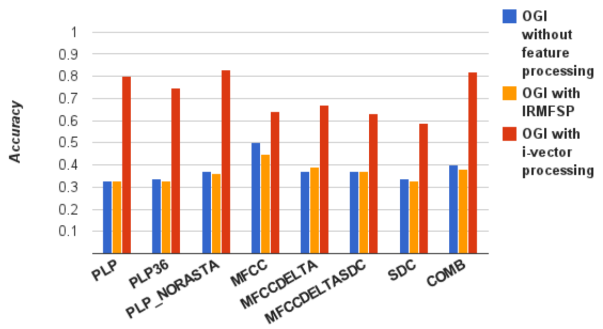

Figure 6: *Results for all feature combinations on the OGIMultilang database, using SVM classifiers.*

OGIMultilang The *OGIMultilang* corpus is bigger and more varied than the *NIST2003LRE* corpus, which makes it harder to classify. As shown, the high-dimensional pure features did not produce good results, with a maximum of 50% for MFCCs. Feature selection using IRMFSP did nothing to improve this result either. I-Vector extraction, however, improved the result by a large margin. Feature-wise, PLP without RASTA processing seems to work best at a result of 83%. MFCC and SDC features did not work quite as well, but did not hurt the result either when combined with PLPs (COMB result). It is interesting to see that the i-vector extraction provided the smallest improvement for MFCCs, the feature that worked best without it.

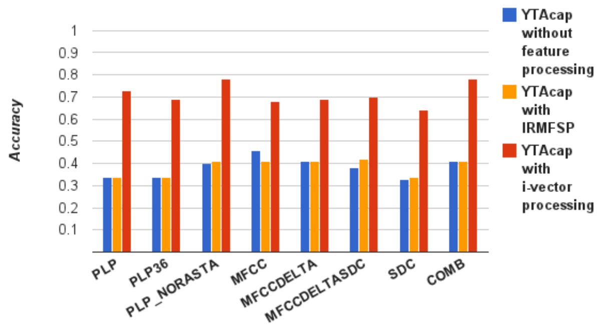

Figure 7: *Results for all feature combinations on the YTAcap database, using SVM classifiers.*

YTAcap Similar to the *OGIMultilang* corpus, the *YTAcap* corpus provides very complex and varied data. We see the same effects on the direct feature training here, too: MFCCs provide the best results, but the accuracy is not very high at just 46%. IRMFSP still does not seem to be able to reduce the feature complexity in a salient way. I-Vector extraction, again, serves to improve the result by a large percentage. The highest result when using i-vector extraction is 78% when using PLP without RASTA processing or the COMB configuration.

6. CONCLUSION

In this paper, we presented an approach for automatic language identification on speech and acapella singing corpora. We used PLP, MFCC, and SDC features, and ran them through an i-vector extractor. We used the generated i-vectors as inputs for MLP and SVM training. To our best knowledge, the i-vector approach is new to music information retrieval. The basic idea behind it is the removal of speaker- and channel-dependent components of the signal. This effectively reduces irrelevance to the language identification tasks and also reduces the amount of training data massively.

Our smallest data set was the *NIST 2003 Language Recognition Evaluation (NIST2003LRE)* corpus. We did not achieve good results for any feature configuration when using the MLP backend. We believe that the small size of the corpus leads to overtraining. I-Vector processing only amplified this problem by reducing the amount of data even further. The SVM backend, however, produced good results of up to 94% for almost all features, with and without i-vector extraction.

The *OGIMultilang* corpus is a much bigger speech corpus. Training without i-vector extraction did not work well for any feature configuration. The best accuracy for this scenario was 52%. Feature selection using IRMFSP did not improve this result. I-Vector extraction, however, improved the results for all feature configurations immensely. We achieved results of up to 83% when i-vector processing was performed. There does not seem to be a large difference between SVM and MLP training, with SVM having just a slight advantage.

We expected language identification for singing to be a harder task than for speech due to the factors described in section 2.2. The results on the *YTAcap* corpus turned out to be somewhat worse than those for the *OGIMultilang* corpus, which is of similar size. We observed the same effect as on *OGIMultilang*: Fairly bad results on the raw features that improved by a large percentage through i-vector extraction. In this case, the accuracy jumped from 56% to 78%.

In general, MLPs seem to work a little better when raw features are used, while SVMs work better when i-vector processing is applied, but only by a small percentage.

Concerning the features, both PLPs and MFCCs seemed to be able to discriminate between languages. PLPs worked best when no RASTA pre-processing was used. We believe that this is because the recordings are all relatively high quality and not heavily spectrally distorted. A higher model order did not significantly increase the accuracy either.

MFCCs worked best when combined with deltas and double-deltas. SDCs by themselves did not work as well as PLP or MFCC features, but were able to increase the accuracy somewhat when combined with MFCCs and their deltas. This confirms our observation mentioned in [12].

The best accuracies were usually achieved when combining MFCCs, MFCC deltas, and PLP features, both features covering different relevant components.

When using an MLP backend, i-vector processing seems to increase the accuracy roughly equally for each feature configuration. Interestingly, this is not true for the SVM backend. In the SVM experiments, MFCCs usually produced the best results when used directly for training. I-Vector extraction provided the smallest improvement for this configuration, but improved the PLP configurations much more.

Overall, i-vector extraction reduces irrelevance in the training data and there leads to a more effective training. As additional benefits, the training process itself is much faster and less memory is used due to its data reduction properties. Using this system, we achieve results that are comparable to the system described in [10] and higher than other publications on the topic of singing language identification. Most of these approaches are based on PPRLM, which requires phoneme-wise annotations and a highly complex recognition system, using both acoustic and language models. In this respect, our system is easier to implement and merely requires language annotations.

7. FUTURE WORK

Since our algorithm produced good results on acapella data, we would now like to test it on polyphonic music. For this purpose, we will integrate additional pre-processing techniques, such as vocal activity detection and source separation.

We will then use the results produced by our language identification algorithm to improve other classification solutions. Genre classification and regional classification are of particular interest here.

In this context, we will expand the music material to different styles, such as opera music or especially non-western music.

We showed that the i-vector extraction algorithm improved our classification accuracy by a large percentage. To our best knowledge, it has not yet been applied to any other Music Information Retrieval problems, such as genre recognition or emotion detection. We are going to investigate these applications as well.

8. REFERENCES

[1] A. Kruspe, H. Lukashevich, J. Abesser, H. Grossmann, and C. Dittmar, “Automatic classification of musical pieces into global cultural areas,” in *Proceedings of Audio Engineering Society 42nd Conference*, Ilmenau, Germany, 2011, pp. 44–53.

[2] J.-J. Aucouturier and F. Pachet, “Improving timbre similarity: How high is the sky?,” in *Journal of Negative Results in Speech and Audio Sciences*, 2004, vol. 1.

[3] M. A. Zissman, “Comparison of four approaches to automatic language identification of telephone speech,” *IEEE Transactions on Speech and Audio Processing*, vol. 4, no. 1, pp. 31–44, Jan. 1996.

[4] Y. K. Muthusamy, E. Barnard, and R. A. Cole, “Reviewing automatic language identification,” *IEEE Signal Procesing Magazine*, vol. 11, no. 4, pp. 33 – 41, Oct. 1994.

[5] E. Singer, P. A. Torres-Carrasquillo, T. P. Gleason, W. M. Campbell, and D. A. Reynolds, “Acoustic, phonetic, and discriminative approaches to automatic language identification,” in *Proceedings of Eurospeech*, Geneva, Switzerland, 2003, pp. 1345–1348.

[6] F. Allen, E. Ambikairajah, and J. Epps, "Language identification using warping and the shifted delta cepstrum," in *2005 IEEE 7th Workshop on Multimedia Signal Processing*, Shanghai, China, 2006, pp. 1–4.

[7] W. M. Campbell, J. P. Campbell, D. A. Reynolds, E. Singer, and P. A. Torres-Carrasquillo, "Support vector machines for speaker and language recognition," *Computer Speech and Language*, vol. 20, pp. 210–229, 2006.

[8] J. Schwenninger, R. Brueckner, D. Willett, and M. E. Hennecke, "Language identification in vocal music," in *7th International Conference on Music Information Retrieval (ISMIR)*, Victoria, Canada, 2006, pp. 377–379.

[9] W.-H. Tsai and H.-M. Wang, "Towards automatic identification of singing language in popular music recordings," in *5th International Conference on Music Information Retrieval (ISMIR)*, Barcelona, Spain, 2004, pp. 568–576.

[10] M. Mehrabani and J. H. L. Hansen, "Language identification for singing," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, Prague, Czech Republic, 2011, pp. 4408–4411.

[11] V. Chandraskehar, M. E. Sargin, and D. A. Ross, "Automatic language identification in music videos with low level audio and visual features," in *Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, Prague, Czech Republic, 2011, pp. 5724–5727.

[12] A. Kruspe, J. Abesser, and C. Dittmar, "A GMM approach to singing language identification," in *AES 53*, London, UK, 2014.

[13] A. Martin and M. Pryzbocki, "2003 NIST Language Recognition Evaluation," Tech. Rep., Linguistic Data Consortium, Philadelphia, 2006.

[14] R. Cole and Y. Muthusamy, "OGI Multilanguage Corpus," Tech. Rep., Linguistic Data Consortium, Philadelphia, 1994.

[15] H. Hermansky, "Perceptual linear predictive (PLP) analysis of speech," *J. Acoust. Soc. Am.*, vol. 57, no. 4, pp. 1738–52, Apr. 1990.

[16] H. Hermansky, N. Morgan, A. Bayya, and P. Kohn, "RASTA-PLP speech analysis," Tech. Rep. TR-91-069, ICSI, 1991.

[17] B. Bielefeld, "Language identification using shifted delta cepstrum," in *Fourteenth annual speech research symposium*, Baltimore, MD, USA, 1994.

[18] 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 *International Conference on Spoken Language Processing (ICSLP)*, Denver, CO, USA, 2002, pp. 89–92.

[19] N. Dehak, P. J. Kenny, R. Dehak, P. Dumouchel, and P. Ouellet, "Front-End Factor Analysis for Speaker Verification," *IEEE Transactions on Audio, Speech, and Language Processing*, vol. 19, no. 4, pp. 788–798, May 2011.

[20] D. Martinez, O. Plchot, and L. Burget, "Language Recognition in iVectors Space," in *Interspeech*, Florence, Italy, August 2011, pp. 861–864.

[21] D. A. Reynolds, T. F. Quatieri, and R. B. Dunn, "Speaker verification using adapted gaussian mixture models," in *Digital Signal Processing*, 2000, p. 2000.

[22] C. Charbuillet, D. Tardieu, and G. Peeters, "GMM Supervector for content based music similarity," 2011, number 1, pp. 1–4.

[23] D. Johnson, "ICSI quicknet software package," http://www.icsi.berkeley.edu/Speech/qn.html, 2004.

[24] G. Peeters and X. Rodet, "Hierarchical gaussian tree with inertia ratio maximization for the classification of large musical instrument databases," in *Proc. of the 6th Int. Conference on Digital Audio Effects (DAFX-03)*, Sept. 2003, pp. 1–6.