Odyssey 2010
The Speaker and Language Recognition Workshop
28 June – 1 July 2010, Brno, Czech Republic

PCA-based Feature Extraction for Phonotactic Language Recognition

Tomáš Mikolov, Oldřich Plchot, Ondřej Glembek, Pavel Matějka, Lukáš Burget and Jan "Honza" Černocký

Brno University of Technology, Speech@FIT, Czech Republic
{imikolov|iplchot|glembek|matejkap|burget|cernocky}@fit.vutbr.cz

Abstract

Phonotactic language recognition is one of major techniques used for automatic recognition of spoken languages. We propose a feature extraction technique based on PCA to be used with SVM-based systems. This technique improves speed of the training, in some cases more than 1000 times, allowing systems to be effectively trained on much larger data sets. Speed-up of the test phase can be even greater, which makes the resulting systems much more useful for processing large amounts of data. We report our results on NIST LRE 2009 task.

1. Introduction

For language recognition (LRE) task, two major approaches proved efficient and complementary in evaluations organized by NIST: acoustic modeling, which relies on short context information, and phonotactic modeling that tries to capture longer patterns in speech [1]. In our work, we investigated discriminative phonotactic models based on support vector machines (SVM), which are reported to be performing better than parallel phone recognition followed by language modeling (PPRLM) approach [2].

The first step in phonotactic-SVM LRE is the feature generation. Phoneme recognizers [3] can be used to produce 1-best strings or lattices from training segments. From lattices, feature vectors with fixed length are constructed. Each element in feature vector is expected N-gram count in given segment. Usually, 3-grams or 4-grams are used as features. For example, if phoneme recognizer has phoneme set of size 40, full feature vector size would be $40^3 = 64\,000$. Note, that each segment is represented by fixed length feature vector, with fixed position for each N-gram feature.

The second step consists of building a discriminative classifier. This usually involves feature normalization and selection of the suitable type of the classifier - most frequently, support vector machines with linear kernel are used. For feature normalization, several approaches were investigated [4]. In our work, we only transform each feature vector element by square root function, which squashes the dynamic range of feature vector components [5]. Selection of optimal trade-off between training error and the margin while training SVMs (parameter C in LIBSVM [6] and SVMTorch [7]) is crucial. The final system then consists of N support vector machines, where N is number of languages - all SVMs are trained in one-versus-all manner and solve two-class decision problem.

Search for optimal parameters can be very time consuming, as it takes large amount of time to train the classifiers. With large number of training segments ($> 10\,000$) and high-dimensional feature vectors ($> 50\,000$), training and test phases become impractically slow (can take days); also, memory requirements can be huge. Several approaches were attempted to overcome this problem, mostly based on feature selection: One possibility is to select features based on their relative frequency, the other approach is to keep only the most discriminative features [5]. We have tested using both feature selection methods, with similar results. In this paper, we report results obtained with feature selection based on relative frequency of N-grams.

The novel approach in context of phonotactic language recognition, presented in this paper, is to include a dimensionality reduction transform in the feature extraction step. It can be seen that the data in the feature vectors are very correlated, as they are generated from lattices that represent several hypothesis, which are generally very similar. In our work, we have used Principal Component Analysis (PCA) for feature extraction to reduce the dimensionality of feature vectors from over 100 000 elements to 100 - 4000 features. This allowed us to train systems much faster (1000 times or more). Without this step, it would not even be possible to train certain systems at all, due to the memory and computational restrictions.

As a performance measurement of our systems, we use NIST-defined average cost performace – C_{avg} [8].

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

2. Task specification

2.1. Data Description

The same data as in our NIST LRE2009 submission [8] were used to train our systems. Our data were separated into two independent subsets, which we denoted TRAIN and DEV. The TRAIN subset contained 23 target languages from NIST LRE 2009 task [9] and had 49 190 segments containing 1572 hours of recordings in total, from which we created smaller subset with 9 810 segments (359 hours of recordings) by limiting the number of segments per language to 500 at most. The DEV subset had 57 languages (including the 23 target ones) and a total of about 63 000 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 standard Continuous Telephone Speech (CTS) databases (CallFriend, Switchboard, OGI etc. - see details in [8]) and Voice of America (VOA) data. The evaluation set - EVAL - was defined by NIST for 2009 LRE evaluation [9].

2.2. System Description

The general architecture of our system is based on our NIST LRE 2009 submission. We used our TRAIN data set to train all frontend subsystems, while we used our DEV data to train the backend and also to test performance. To keep backend training and test separate, we resorted to a jackknifing scheme.

Duration independent backend performs fusion and calibration. The backend fuses the scores from the frontends and outputs calibrated scores, which function as multiclass log-likelihoods [8].

All individual frontend subsystems are based on three phoneme recognizers: two left-context/right-context hybrids and one based on GMM/HMM context dependent models.

2.3. Hybrid Phoneme Recognizers

The phoneme recognizer is based on hybrid ANN/HMM approach, where artificial neural networks (ANN) are trained to produce emission probabilities for HMM states from Mel filter bank log energies using the context of 310ms around the current frame. The expected N-gram phoneme counts estimated from lattices form the feature vectors [10]. Hybrid recognizers were trained for Hungarian and Russian on the SpeechDat-E databases. For more details see [11, 3]

2.4. 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 [12]. The models were trained using 2000 hours of English CTS 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 Heteroscedastic Linear Discriminant Analysis (HLDA) transformation. The models are trained discriminatively using Minimum Phone Error (MPE) criterion [13]. VTLN and MLLR adaptation are used for both training and recognition in SAT fashion. The triphones were used for phoneme recognition with a bigram phonotactic model trained on English-only data.

In all subsystems, the expected N-gram phoneme counts from corresponding phoneme recognizer were used for subsequent classification by SVM, similarly to MIT's work [14].

The number of phonemes for each system was different, and so was the feature vector size. Hungarian recognizer used phoneme set of 33 phonemes, English recognizer used 46 and Russian 53.

3. Feature Selection

We started to build our system by using 3-gram and 4-gram features generated by Hungarian recognizer (HU3 and HU4 systems). These features are square roots of expected N-gram counts from lattices. The phoneme set size of HU recognizer was 33 phonemes, resulting in $33^3 = 35\,937$ possible features for 3-gram system and $33^4 = 1\,185\,921$ for 4-gram system. We have used 9 810 training segments out of the 49 190 training segments available (the small TRAIN set, see above).

Direct training of SVM models based on huge amount of data can be often intractable. In the context of phonotactic language recognition, two methods of feature selection, for reducing the dimensionality of the final feature vector, are usually used: feature selection based on relative frequency, which discards N-grams with low frequency, or discriminative feature selection, which discards the least discriminative N-grams [5].

We have used feature selection based on the relative frequency only for the HU4 system - see table 1. The results indicate that it is useful to keep as many features as possible. However, memory and time complexity raises significantly - it took several days to train the system using feature vectors containing 80 000 components. Also, the testing takes large amount of time, making these models impractical for real applications.

4. Feature Extraction based on PCA

To increase the effectiveness of the SVM phonotactic approach, we investigated feature extraction[1] with popular Principal Component Analysis. We perform the feature

[1] By the feature extraction in this context, we mean the dimensionality reduction, which produces a new set of smaller feature vectors.

Table 2: Dimensionality reduction for HU3 system from 35 937 features, times are in seconds. Systems marked with * were trained on a slower computer (approximately twice) because of memory demands. Note that time to project data (6th column) involves time needed to project both training and test data.

Reduction	DEV Cavg (30s)	Training(s)	Testing(s)	Computing PCA(s)	Projecting data(s)	Total time(s)	Speedup
$\rightarrow$ 100	2.83	93	75	104	22	294	1 080
$\rightarrow$ 500	2.43	423	407	658	108	1 596	199
$\rightarrow$ 1 000	2.38	884	946	2 609	220	4 659	68
$\rightarrow$ 2 000	2.32	2 110	2 289	11 099	399	15 897	20
$\rightarrow$ 4 000	2.28	4 296	4 848	93 110*	1 743*	103 997	3.05
no reduction	2.33	124 565*	193 168*	-	-	317 733*	1.0

Table 1: Performance of HU4 system with feature selection on DEV data.

feature size	DEV Cavg 30s
5 000	4.0
10 000	3.5
20 000	3.0
40 000	2.8
80 000	2.7

extraction step after applying the square root compression and feature selection (if it is performed). First, we create matrix $\mathbf{M}$, where each row contains one feature vector. For HU3 system based on 35 937 features, this means that $\mathbf{M}$ has size $35\,937 \times 9\,810$. Next, we compute some amount (typically 500-2000) of the most important principal components. These are then used to project data to low-dimensional space. SVMs are then trained on these "compressed" data.

To implement the system, we have used LIBSVM toolkit [6][2]. We used the linear kernel and tuned the parameters on the DEV set. To compute principal components, we used an implementation of Randomized algorithm for principal component analysis [15].

The effect of this dimensionality reduction approach can be seen in Table 2. As our goal is also to increase the speed of the system, we report times to compute principal component analysis, to project training and test data to low-dimensional space, to train systems and to test them on the DEV data using only 30s utterances (13331 segments).

Our approach thus involves the following steps in the training phase:

1. Compute square root of expected N-gram counts from each training segment
2. Select appropriate features (most frequent N-grams)
3. Create matrix $\mathbf{M}$
4. Compute the most important principal components
5. Extract new feature vectors from training data set using these principal components
6. Train SVMs on these reduced feature vectors

In the test phase, we follow these steps:

1. Compute square root of expected N-gram counts from each testing segment
2. Select the same features as in the training phase
3. Extract new feature vectors from test data using principal components computed from the training data
4. Test SVMs on these reduced feature vectors

For example, if we reduce system with 35 937 original features to a system with 500 features as shown in Table 2, the training phase takes 658+(108/2)+423 seconds, while testing phase takes (108/2)+407 seconds.

The results indicate that the feature dimensionality reduction from 35 937 to 500 features provides 109 times speedup of the training phase. The test phase is 419 times faster, with only small degradation in accuracy. Reduction to just 100 features results in significant decrease of accuracy, while speedup of the test phase is more than 2200. On contrary, by reducing feature size to 4000, the accuracy is slightly improved, and the overall performance is better than that of the original system. However, this small improvement in accuracy is not very interesting and is of small practical importance. Our main motivation in this work was to speed up training and test phase to allow systems to be trained on much larger data sets.

As PCA does not need to be estimated from the whole data set, it is also possible to reduce the computational time by using only a subset of the data for PCA estimation. Preliminary results indicate that it is possible to do so with only minimal degradation of accuracy (however, it is needed to equalize the amount of data from all languages).

[2]To improve speed, we used *-m 4000* option to allocate 4 GB of cache.

Table 3: Performance of systems trained on feature vectors of different size. All systems were trained on 9810 segments, except RU3-ALL which was trained on 49190 segments (whole TRAIN set).

system	features	reduction	EVAL		
			3s	10s	30s
HU3	3-gram	35 937 → 500	21.65	9.29	4.0
HU3	3-gram	35 937 → 1 000	21.58	9.21	3.86
HU3	3-gram	35 937 → 4 000	21.51	9.18	3.85
HU4	4-gram	80 000 → 1 000	22.38	9.75	4.09
EN3	3-gram	63 600 → 500	22.16	9.13	3.50
EN3	3-gram	63 600 → 1 000	22.30	9.17	3.48
EN4	4-gram	100 000 → 500	25.12	9.67	3.64
RU3	3-gram	115 400 → 2 000	20.11	7.76	3.26
RU4	4-gram	150 000 → 500	20.16	7.62	3.37
RU3-ALL	3-gram	115 400 → 1 000	19.20	6.82	3.03

5. Results with multiple systems

Next, we have trained various systems based on the other phoneme recognizers - English and Russian. Table 3 summarizes results achieved on all conditions used in NIST LRE 2009. The results have been reported after duration-independent calibration (see [8] for details). It can be seen that for EN3 system, the feature reduction to 500 dimensions is sufficient - for each phoneme recognizer, the optimal size should be determined on the development set. It is interesting to see that by using more training data, it is possible to obtain significant improvement (by comparing RU3 system trained on 9810 segments and RU3-ALL system trained on 49 190 segments). As RU3-ALL system uses 115 400 features before reduction, it would be impractical to train such system without dimensionality reduction step in feature extraction.

Table 4 summarizes the results of our systems after fusion [8] - we have used HU3-4000, EN3-1000, RU3-2000 trigram systems and HU4-1000, EN4-500, RU4-500 4-gram systems. It is interesting to see that all systems based on 4-gram features are contributing very little after fusion with trigram systems. One single system trained on all available data - RU3-ALL-1000 - seems to provide considerable improvement after fusion with the other systems.

In our other work[16], we were conducting experiments with calibration, different composition of development set and detecting overlapping speakers in the training and development set. We achieved additional significant improvements for all test conditions. The C_{avg} performance of fusion of 13 different SVM systems was **1.78, 3.86 and 14.13** for 30, 10 and 3 second condition.

Table 4: Fusion of multiple systems

fusion of systems	EVAL 3s	EVAL 10s	EVAL 30s
all 3-gram	15.13	5.01	2.39
all 4-gram	15.85	5.0	2.56
3+4-gram	14.94	4.77	2.34
3+4-gram + RU3-ALL	14.77	4.65	2.25

6. Conclusion

In our work, we have shown that using dimensionality reduction as a feature extraction step for phonotactic language recognition based on SVM can lead to very efficient systems. Large speedup can be obtained when training systems on more features and segments. The speed of the test phase can allow use of these systems in real world applications. In some cases, we have achieved more than 1000 times speedup in comparison to systems trained without the feature extraction with minor deterioration of accuracy.

Although the feature extraction based on PCA is simple, it proved to be very efficient. However, future work can explore more advanced dimensionality reduction techniques, such as ICA. As the amount of data we are dealing with is huge, this might not be easy, on the other hand, we expect further improvements.

7. References

[1] Ondřej Glembek, Pavel Matějka, Lukáš Burget, and Tomáš Mikolov, "Advances in phonotactic language recognition," in *Proc. Interspeech*, 2008, p. 4.

[2] Christopher White, Izhak Shafran, and Jean-Luc Gauvain, "Discriminative Classifiers for Language Recognition," in *Proc. ICASSP*, 2006, pp. 213–216.

[3] Petr Schwarz, Pavel Matějka, and Jan Černocký, "Towards lower error rates in phoneme recognition," in *Proceedings of 7th International Conference Text, Speech and Dialoque*, 2004.

[4] A. Stolcke, S. Kajarekar, and L. Ferrer, "Nonparametric feature normalization for svm-based speaker verification," in *Proc. ICASSP*, 2008, pp. 1577–1580.

[5] F.S. Richardson and W.M. Campbell, "Language recognition with discriminative keyword selection," in *Proc. ICASSP*, 2008, pp. 4145–4148.

[6] Chih-Chung Chang and Chih-Jen Lin, "LIBSVM: a library for support vector machines," http://www.csie.ntu.edu.tw/˜cjlin/libsvm, 2001.

[7] Ronan Collobert and Samy Bengio, "SVMTorch: Support Vector Machines for Large-Scale Regression Problems," in *Journal of Machine Learning Research, vol 1*, 2001, pp. 143–160.

[8] Niko Brümmer, Lukáš Burget, Ondřej Glembek, Valiantsina Hubeika, Zdenek Jančík, Martin Karafiát, Pavel Matějka, Tomáš Mikolov, Oldřich Plchot, and Albert Strasheim, "But system description for nist lre 2009," in *Proc. 2009 NIST Language Recognition Evaluation Workshop*. 2009, pp. 1–7, National Institute of Standards and Technology.

[9] "The 2009 NIST Language Recognition Evaluation Plan," http://www.itl.nist.gov/iad/mig/tests/lre/2009/LRE09_EvalPlan_v6.pdf.

[10] J.L. Gauvain, Messaoudi A., and Schwenk H., "Language recognition using phone lattices," in *Proc. ICSLP 2004*.

[11] Petr Schwarz, Pavel Matějka, and Jan Černocký, "Hierarchical structures of neural networks for phoneme recognition," in *Proceedings of ICASSP*, 2006, pp. 325–328.

[12] T. Thomas, V. Wan, L. Burget, M. Karafiát, J. Dines, J. Vepa, G., Garau, and M. Lincoln, "The AMI System for the Transcription of Speech in Meetings," in *Proc. ICASSP 2007*, 2007, pp. 357–360.

[13] D. Povey, "Discriminative Training for Large Vocabulary Speech Recognition," Ph.d. thesis, Cambridge University, July 2004.

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

[15] Vladimir Rokhlin, Arthur Szlam, and Mark Tygert, "A Randomized Algorithm for Principal Component Analysis," Technical report, University of California, 2008.

[16] Jančík Z., Plchot O., Brümmer N., Burget L., Glembek O., Hubeika V., Karafiát M., Matějka P., Mikolov T., Strasheim A., and Černocký J., "Data selection and calibration issues in automatic language recognition investigation with but-agnitio nist lre 2009 system," in *submitted to Proc. Odyssey*, July 2010.