Improved Language Identification Using Support Vector Machines for Language Modeling

Xi Yang [1], Lu-Feng Zhai [1], Manhung Siu [1] and Herbert Gish [2]

[1]Dept. of Electrical and Electronic Engineering, Hong Kong University of Science and Technology
[2] BBN Technologies
yangxi,eelfzhai,eemsiu@ust.hk, hgish@bbn.com

Abstract

Automatic language identification (LID) decisions are made based on scores of language models (LM). In our previous paper [1], we have shown that replacing n-gram LMs with SVMs significantly improved performance of both the PPRLM and GMM-tokenization-based LID systems when tested on the OGI-TS corpus. However, the relatively small corpus size may limit the general applicability of the findings. In this paper, we extend the SVM-based approach on the larger CallFriend corpus evaluated using the NIST 1996 and 2003 evaluation sets. With more data, we found that SVM is still better than n-gram models. In addition, back-end processing is useful with SVM scores in CallFriend which differs from our observation in the OGI-TS corpus. By combining the SVM-based GMM and phonotactic systems, our LID system attains an ID error of 12.1% on NIST 2003 evaluation set which is more than 4% (25% relatively) better than the baseline n-gram system.

Index Terms: language identification, support vector machine, discriminative training, language modeling.

1. Introduction

Class-specific language models (LMs) play an important role in many speech-related classification tasks, such as topic, utterance, speaker or language identification (LID). Traditionally, these language models are trained using maximum likelihood (ML) related criterion. Recently, discriminative model training techniques have been applied with good results [2].

One approach to train discriminative LMs is the use of support vector machines (SVMs) [1]. SVMs are powerful classifiers that have been shown to perform well in many pattern classification problems [3]. In addition to being discriminatively trained, its training criterion balances the reduction of training errors and its generalizability to unseen data. Furthermore, the criterion is convex that can be optimized by quadratic programming techniques. SVMs were proposed for speaker identification in [4, 5, 6] and for count data in [5, 7] for topic and speaker identification.

In our previous paper, we applied SVMs to estimate class-dependent language models for LID on the OGI-TS corpus and studied the selection of kernels, and handling of prior mis-matches and SVM score normalization. However, the relatively small amount of data in that corpus limits the general applicability of the findings. In this paper, we examine the effectiveness of SVM for LID on the larger CallFriend corpus. The larger amount of training data and the availability of a development set allow us to examine in detail the effect of back-end processing, fusion of systems as well as similarity and differences between the GMM tokenization and the PPRLM approaches.

The rest of the paper is organized as follows. In the next section, we briefly review the SVMs and how they are related to LM-based classification problems. Then, we describe our baseline LID systems in Section 3. Section 4 describes how we applied SVM in the GMM-tokenization and PPRLM frameworks. The paper is concluded with a discussion in Section 5.

2. Support Vector Machines

By minimizing structural risks, SVMs balance training errors and generalizability. Denote a set of n-dimensional training samples as $x_i \in R^n$ with class labels $y_i \in \{-1, 1\}$. Consider the separable two-class problem. A linear classifier, $f(x)$, defined using the classification hyperplane as $f(x) = w.x + b$, separates the classes if

$$f(x) > 0 \quad \text{if} \quad y_i = 1 \tag{1}$$

$$f(x) < 0 \quad \text{if} \quad y_i = -1 \tag{2}$$

For SVMs, the optimal hyperplane w is one that separates the classes while maximizing the distance between the hyperplane and the nearest data points. This distance is called the margin which is related to the generalizability of the classifier. It turns out that only a subset of training samples that are close to the hyperplane affect the location of the hyperplane. These are called the support vectors, denoted as $\tilde{x}_i$. The classification hyperplane can be efficiently learned by constrained optimization using quadratic programming techniques. Because the optimization function is convex, global maximal can be obtained.

Non-separable cases can be solved by introducing an extra penalty for errors. For the non-separable case, there is a trade-off between the training classification errors against the size of the margin. With a larger margin, the classifier is better in generalizing to unseen data.

One of the main power of SVM is in its ability to generalize into a non-linear classifier. This is achieved by mapping the features to a high dimensional space, denoted as $\mathcal{H}$, using non-linear functions. If $\mathcal{H}$ is created with complex non-linear mapping functions, calculating the hyperplane can be difficult. Instead, the classifier, $f(x)$, can be written as a function of the inner product between the data and the support vectors. That is,

$$f(x) = \sum_{i}^{N_s} \alpha_i K(x, \tilde{x}_i) + b,$$

This work is partially supported by the Hong Kong Research Grant Council under CERG grant number HKUST 6210/03E.

where N_s is the number of support vectors, α_i is the weight for the support vectors $\tilde{x}_i$ and $K(.,.)$ is the **kernel** function defining the inner product between x and $\tilde{x}_i$. By using kernels, the space $\mathcal{H}$ in fact does not need to be explicitly defined. Some commonly used kernels include the polynomial kernels and the Gaussian radial basis function kernels. For details about SVMs, readers can refer to [3, 8]

2.1. LM-based Classification as a Linear Classifier

In language identification, classification is mainly decided based on the token patterns captured by class-specific language models. Denote $c_l(i, j)$ as the count in class l, of observing token i, j consecutively, and denote $c_l(i)$ as the count of token i. The bigram probability (via interpolated n-gram), $p_l(j|i)$, is given by

$$p_l(j|i) = (1-\alpha)\frac{c_l(i,j)}{c_l(i)} + \alpha\frac{c_l(i)}{\sum_i c_l(i)}$$

where α is the linear interpolation parameter to smooth the bigram with unigram. Higher order n-gram model probabilities can also be estimated in a similar fashion.

Using above model for classification can be viewed as deriving a separating hyperplane from the n-gram probabilities. To simplify our discussion, consider deciding between two classes, a and b and arrange the n-gram counts in a vector from, denote as x with $x[i]$ being the i-th n-gram. Similarly, the n-gram probabilities, denoted as p_a and p_b for the two classes can also be arranged in vector form. By labeling class a as class "1" and b as class "-1", a traditional n-gram model classifier would have the separating hyperplane $\hat{w}$ given as

$$\hat{w}[i] = \log\frac{p_a[i]}{p_b[i]}.$$

This shows that the n-gram is a special case of a linear classifier with the n-gram counts as the features and the log odds as the weights, which can be replaced by discriminatively trained SVMs.

While SVMs are mostly defined for two-class classification problems, several approaches have been suggested in the literature to perform n-way classification. These include building binary classifiers between all class-pairs, one-against-all classifiers for each target class and, using a coding approach [9]. Since there is no clear performance advantage of using one approach versus the other, we use the one-against-all approach because of its simplicity.

3. Baseline LID system

3.1. LID System Overview

Most LID systems consist of a set of frontend tokenizers to convert speech acoustics into sequences of discrete tokens. Two widely used tokenization approaches are: the phoneme recognizer [10], and the Gaussian mixtures [11]. Other tokenizers, such as syllables [12], were also proposed. During training, language patterns are captured by class-specific language models. During test, language identification decisions are made by comparing the language modeling likelihoods or scores. The language modeling scores are typically transformed for better classification by the *backend processing* (*bp*). Scores from different tokenizers or systems are then combined by *fusion* to make a single classification decision.

3.2. Baseline Experiment Setup

Our LID experiments were performed on the CallFriend Corpus and evaluated on both the NIST 96 and NIST 2003 evaluation sets. In this work, we focused on the close-set LID instead of verification. Twelve languages were identified including: American English, Arabic, Farsi, Canadian French, Mandarin, German, Hindi, Japanese, Spanish, Korean, Tamil and Vietnamese. The training consisted of 20 complete 30 minutes conversations for each of the 12 languages. We performed experiments only on the 30-second test subset. The NIST 1996 development (*dev96*) and evaluation set (*eval96*) consists of 1147 and 1492 segments respectively. The 2003 evaluation set (*eval03*) consists of 1280 utterances of which 960 come from CallFriend. Furthermore, 80 of the non-CallFriend utterances are Russian and were excluded in our experiments. The *dev96* and *eval96* together serve as the development data for the *eval03*.

Two separate systems were built, one using the GMM-tokenization approach and another the PPRLM approach. These systems were fused to form the final system.

For the GMM tokenization approach, the baseline system used 2048 GMM components. 12 GMM-tokenizers were trained, one for each language, using language specific data. Shifted delta cepstral coefficients with configuration 7-1-3-7 [13] were used as the acoustic features. During training, silence detection was performed with a two-state GMM-based silence detector together with energy information from both sides of a conversation. Silence detection was also applied during test. Interpolated LMs were used in the baseline experiments with the weights of 0.66, 0.33 for unigram and bigram [11]. This system is denoted as the GMM-LM system.

The PPRLM system consisted six different phoneme recognizers, which are: English, German, Hindi, Japanese, Mandarin and Spanish. They were trained from the OGI-TS corpus and each phoneme was represented by a 3-state, left-to-right hidden Markov model with 4 Gaussian mixtures per state. These phoneme recognizers have a phoneme recognition accuracy of approximately 35% on the OGI-TS corpus. To improve the recognizers' performance on the Callfriend Corpus, unsupervised MLLR adaptation with 2 regression classes were performed but only on a small subset of the training data. Interpolated trigram language models were used, trained on the recognized phoneme sequences.

Two *backend processing* and fusion schemes were tested. In the first case, denoted as *backend-processing-with-weighted-average* (*bp-wt-ave*) [10], applied the *backend processing* before combining scores across tokenizers. For each test sequence from a single tokenizer (applicable to both the GMM-LM and PPRLM), scores from the LMs were stacked into a 12-dimensional vector and transformed by a linear discriminative analysis (LDA) matrix that reduced the data vector dimension to 11. The LDA matrix was trained on the respective development set. The resultant 11-dimensional vector was then considered as features in a Gaussian classifier which used a single Gaussian with diagonal covariance to represent each language. While the likelihood of the Gaussian classifier could be used to make LID decision per tokenizer, they were again considered as scores and were linearly combined with scores from other tokenizers. The weights in this linear combination were estimated from the development set.

For the GMM-LM system, in addition to the transformed LM scores from the 12 tokenizers, the acoustic scores from the 12 GMM tokenizers can also be used and considered as a separate systems. Thus, the fusion involved the weighted combination of

	eval96		*eval03*	
	GMM	PPRLM	GMM	PPRLM
before *bp*	34.3	66.7	33.2	56.3
after *bp*	61.3	66.8	54.0	60.3
bp-wt-ave (w/o *Ac-sc.*)	70.4	83.2	63.3	77.7
fn-bp (w/o *Ac-sc.*)	**71.8**	**84.2**	**70.3**	**78.3**
bp-wt-ave (w *Ac-sc.*)	75.3	-	65.9	-
fn-bp (w *Ac-sc.*)	**74.7**	-	**75.6**	-

Table 1: Baseline LID accuracy of PPRLM and GMM-LM with two types of back-end/fusion processing

13 systems. For the PPRLM system, scores from the 6 phoneme recognizer were combined.

Alternatively, scores from all LMs and all tokenizers can be first stacked into a single vector, then transformed by an LDA matrix and evaluated by a Gaussian classifier. This approach is denoted as *fusion-before-backend-processing* (*fn-bp*). Using this approach, the GMM-LM system, including the acoustic likelihood, will have a stacked vector of $13 \times 12 = 156$ and the LDA matrix of dimension 156×11. For the PPRLM, the stacked vector will have the dimension of $6 \times 12 = 72$ and the LDA matrix of dimension 72×11.

3.3. Baseline Results

Baseline results are tabulated in Table 1. The top two lines show the average single tokenizer performance before and after backend processing (*bp*). Significant improvement was obtained for both the GMM-bigram and PPRLM-trigram systems after *backend processing* and fusion. For the GMM-based system, including the acoustic likelihood score (denoted as *Ac-sc.*) further improved performance. Note that even though there is a mis-match between the OGI-TS trained phoneme recognizers and CallFriend, the PPRLM system still outperformed the GMM-LM system. Weighted combination of the GMM-LM and PPRLM scores (denoted in **bold** above) resulted in an accuracy of 86.7% and 83.3% for *eval96* and *eval03* respectively which are 3 to 5% (absolute) better than the PPRLM results as shown in Table 4.

4. SVM-based System

As discussed in Section 2, the ML estimated n-grams can be replaced by SVMs with frequency of n-token sequences as features. In our previous paper [1], we found that the balance of prior is very important to SVM performance. Furthermore, back-end processing contributed little to the overall performance and simple averaging performed the best over other fusion techniques. Some of these findings can be affected by the amount of test and development data. In addition, with large amount of training data, modifications to the SVM training will be needed especially for the GMM-tokenizer sequence to reduce the possible number of features. Similar to our work in [1], all SVM experiments used a linear kernel with the default settings with the inverse-document-frequency (IDF) weighting on the SVM features. Our preliminary experiments on using other kernels or with different settings do not resulted in any significant improvements.

	eval96		*eval03*	
	1-gm	2-gm	1-gm	2-gm
before *bp*	65.1	62.3	57.0	53.0
after *bp*	66.1	62.8	60.4	57.9
bp-wt-ave (w/o *Ac-sc.*)	75.4	73.7	71.7	69.7
fn-bp (w/o *Ac-sc.*)	71.7	73.3	73.9	72.3
bp-wt-ave (w/ *Ac-sc.*)	**77.6**	76.2	71.8	71.2
fn-bp (w/ *Ac-sc.*)	75.7	75.9	**76.8**	76.2

Table 2: LID accuracy of GMM-SVM with two types of back-end/fusion processing

	eval96		*eval03*	
	2-gm	3-gm	2-gm	3-gm
before *bp*	72.3	73.7	64.3	64.9
after *bp*	71.1	74.4	65.2	70.0
bp-wt-ave	88.1	88.8	81.9	84.1
fn-bp	85.1	**89.8**	81.2	**84.8**

Table 3: LID accuracy of PPRSVM with two types of back-end/fusion processing

4.1. GMM-SVM System

We call the system with GMM tokenizers as the GMM-SVM system. Because the tokenizer has 2048 mixtures, the number of possible token n-grams is huge. For bigram, it can be as high as 2048^2. To make this easier on the SVM, feature reduction is needed. A simple way to reduce the number of n-grams is to set a minimum count threshold. An alternative is to train a different GMM tokenizer with smaller size just for higher order n-gram. For example, we can use a 2048 component GMM for unigram modeling but a 512-component GMM for bigram modeling.

In our experiments, we combined the above two approaches. For bigram, the 2048-component GMM was used but with a minimum count of 70. This threshold was found to give the best balance between performance and feature counts. For trigrams, because the 2048-component GMM would create too many distinct trigram, we used a 256-component GMM instead, together with the minimum count constraint.

Table 2 shows the GMM-SVM performance with different n-gram orders. We notice that unigram and bigram have very similar performance and that *fusion-before-backend-processing* is significantly better than *bp-wt-ave* for *eval03*. Comparing the performance in Tables 1 and 2, before the addition of the acoustic scores, GMM-SVM is significantly better than GMM-LM by more than 3% in both *eval96* and *eval03*. However, after the addition of the acoustic scores, the benefit of SVM is vastly decreased to less than 1%.

4.2. PPR-SVM System

Similar to the GMM-SVM system, SVMs can replace the n-gram models in PPRLM. We called this the PPRSVM system. The results are shown in Table 3. Compared to the results in Table 1, the PPRSVM is more than 5% better than the PPRLM irrespective of which fusion approach was applied. Different from GMM-SVM, there is little difference between *fusion-before-backend-processing* and *backend-processing-with-weighted-average*. Similar to PPRLM, the performance on *eval96* is more than 6% better than *eval03*.

	eval96		*eval03*	
	wt-ave	*fn-bp*	*wt-ave*	*fn-bp*
GMMLM + PPRLM	**86.7**	82.8	**83.3**	81.3
GMM-SVM + PPRSVM	88.9	**88.5**	86.1	**87.9**

Table 4: LID accuracy of combining GMM-SVM and PPRSVM

4.3. Combination of SVM-based systems

The GMM-SVM and PPRSVM systems can be combined. Because the PPRSVM system is significantly better than the GMM-SVM system, the potential gain in this case, may be smaller than in other combinations. We again tried two combination approaches. In the first one, the language scores are linearly combined with the weights learned from the development set. Alternatively, we consider the two systems as two frontends and applied the *fusion-before-backend-processing* above. The results of fusion of GMM-LM and PPRLM, and fusion of GMM-SVM and PPLSVM are tabulated in Table 4. Combination of the PPRSVM and GMM-SVM systems gives good improvement for *eval03* but degraded performance slightly in *eval96*. That is probably because of the larger difference in performance between GMM-SVM and PPRSVM in *eval96* and that the backend processing parameters and the system combination weights were both estimated from the same development set which can cause over-tuning on the set. Interestingly, out of the two combination schemes tested, the baseline system performs better when using linear combination while the systems perform slightly better with another level of *fusion-before-backend-processing*.

5. Discussion

The experimental results demonstrate the usefulness of SVM as replacement of language models for language identification even when applied to a large corpus such as CallFriend. This is especially evident in the single tokenizer results before backend processing. Absolute gains of 20-30% and 6-7% are observed in GMM and phonotactic based systems respectively.

Backend processing improves the baseline GMM and phonotactic systems by 20-27% and 1-4% respectively but SVM performance can only be improved slightly (1-3% and 2-5%). The fact that backend processing is useful for SVM is different from our findings in [1] probably because of the availability of development set for learning the *backend processing* parameters in CallFriend. A smaller gain from *backend processing* is expected because SVM is discriminatively trained and thus, the benefit of a discriminatively trained LDA may become smaller.

Both fusion approaches give good improvement and there is no clear winner as to which one is better. There seems to be interaction between the evaluation set and type of system. However, in most cases, the *fusion-before-backend-processing* gives better performance on *eval03* and comparable performance on *eval96* while there is much bigger variance in the performance of *backend-processing-with-weighted-average*. Different from our findings in [1], simple average after *backend processing* is not as good as weighted averages which can be explained by the better weight estimation using the development set.

For the GMM-LM, the addition of acoustic scores gives a gain of 4-5% but for the GMM-SVM, the gain from adding the acoustic scores is much smaller in the range of 2-3%. It is not clear what correlations there is between the SVM and the acoustic scores other than potentially dynamic range issues. This also reduces the gain of GMM-SVM over GMM-LM. Because the GMM-SVM system uses a very large feature set, there may also be issues in convergence or balance between SVM training margins. These issues will require further study.

From Tables 1, 2 and 3, one can notice many differences in behavior, such as the selection of fusion scheme, between *eval96* and *eval03*. There are 3 major differences between the two evaluation sets. They are: 1) overlap in *dev96* and *eval96*, 2) non CallFriend data in *eval03*, and 3) larger amount of development data for *eval03*. Whether any of these causes the difference in behavior is not clear and further study is needed.

Overall, comparing the best combination of the baseline systems with the best combination in the SVM-based systems, the SVM systems outperforms by 2-4%. More combination is also possible, including the combination of different n-gram of different orders, either in SVM features or in post processing. Our preliminary results are very encouraging with improvement on the *eval03* closed to 3% absolute. This is currently being investigated.

6. References

[1] Zhai, L. Siu, M., Yang, X, and Gish, H., "Discriminatively trained Language Models Using Support Vector Machines for Language Identification", to appear in Odyssey 2006

[2] Stolcke, A. *et al*, "The SRI March 2000 Hub-5 conversational speech transcription system", In Proc. NIST speech Transcription Workshop 2000

[3] Vapnik, V., "The nature of statistical learning theory", Springer-Verlag, New York, 1995.

[4] Schmidt, M. and Gish, H., "Speaker identification via support vector machines", in Proc. ICASSP 1996, pp 105-108.

[5] Campbell, W. M. *et al*, "Phonetic Speaker recognition with support vector machines", In Proc. NIPS, pp 1377-1384, 2003.

[6] Campbell, W. M. *et al*, "Support vector machines for speaker and language recognition", Computer, Speech and Language, Vol 20, 2006, pp. 210-229.

[7] Belfield, W., and Gish, H., "A topic classification system based on parametric trajectory mixture models", In Proc. Eurospeech, pp 1269-1272, 2003.

[8] Burges, C., "A tutorial on support vector machines for pattern recognition", Data Mining and Knowledge Discovery, 2(2), 1998.

[9] Scholkopf, B. and Smola, A., "Learning with Kernels: Support vector machines, regularization, optimization, and beyond", The M.I.T. press, 2002.

[10] Zissman, M., "Comparison of Four Approaches to Automatic Language Identification of Telephone Speech", IEEE Trans. of Sp. & Audio. Proc., vol. 4, 1996, pp. 31-44.

[11] Torres-Carrasquillo P. *et al*, "Language Identification using Gaussian Mixture Model Tokenization", ICASSP 2002

[12] Nagarajan, T., and Murthy, H., "Language identification using parallel syllable-like unit recognition", in Proc. ICASSP 2004, pp. 401-404.

[13] Torres-Carrasquillo P. *et al*, "Approaches to Language Identification using Gaussian Mixture Models and Shifted Delta Cepstral Features", ICSLP 2002, pages 89-92