LANGUAGE RECOGNITION WITH DISCRIMINATIVE KEYWORD SELECTION*

F. S. Richardson, W. M. Campbell

MIT Lincoln Laboratory

ABSTRACT

One commonly used approach for language recognition is to convert the input speech into a sequence of tokens such as words or phones and then to use these token sequences to determine the target language. The language classification is typically performed by extracting N-gram statistics from the token sequences and then using an N-gram language model or support vector machine (SVM) to perform the classification. One problem with these approaches is that the number of N-grams grows exponentially as the order N is increased. This is especially problematic for an SVM classifier as each utterance is represented as a distinct N-gram vector. In this paper we propose a novel approach for modeling higher order N-grams using an SVM via an alternating filter-wrapper feature selection method. We demonstrate the effectiveness of this technique on the NIST 2005 language recognition task.

***Index Terms*—** Language Recognition, Support Vector Machines

1. INTRODUCTION

Language recognition systems generally fall into two categories: acoustic systems which model short term spectral characteristics of speech and token systems which model *high level* features such as words or phones. State-of-the-art systems typically include both techniques to achieve optimal performance. In this paper, we focus on a token based language recognition system which uses a phone recognizer for generating the token sequences and a support vector machine for classifying the target language.

Token-based SVM systems have been used in many speaker and language recognition systems [1, 2, 3]. Sequence kernels are constructed by viewing a speech segment as a *document* of tokens. The SVM feature space in this case is a scaling of co-occurrence probabilities (or N-grams) of tokens in an utterance. This technique is analogous to methods for applying SVMs to text classification [4].

In our prior work [5], we used the output of a speech recognition system to generate sequences of words. We observed that certain words (or keywords) were strongly indicative of a specific language. For example, the phrase “you know” was a strong indicator that the language was English. One limitation with this approach was that the possible set of words for distinguishing between languages was limited to the word lexicon of the under lying speech recognition system.

In this paper, our token sequences are generated using a phone recognizer. We then attempt to find keywords by selecting the N-grams which discriminate the most between the different languages for progressively higher orders of N. Unfortunately, this problem is difficult to solve directly, since the number of unique N-grams grows exponentially as we increase the order. To alleviate this difficultly, we propose a method that starts by selecting a relatively small set of lower order N-grams and then using these sets to successively build candidate lists of higher order N-grams.

The outline of the paper is as follows. In Section 2.1, we review the basic architecture that we use for phone recognition and how it is applied to the problem. In Section 2.2, we review the application of SVMs to determining a target language. Section 3.1 describes a feature selection method for SVMs. Section 3.2 presents our method for constructing long context units of phones to automatically create keywords. We use a novel feature selection approach that attempts to find longer strings that discriminate well between classes. Finally, in Section 4, we show the application of our method to a language recognition problem. We show qualitatively that the method produces interesting keywords. Quantitatively, we show that the method produces keywords which are good discriminators between classes.

2. PHONOTACTIC CLASSIFICATION

2.1. Phone Recognition

The high-level component of our system is a phone recognizer based upon the Brno University (BUT) design [6]. The basic architecture of this system is a three state monophone HMM system with a null grammar. This system uses two powerful components to achieve high accuracy. First, TRAPS [7] which are long time-span time-frequency features provide contextual cues for modeling monophones. Second, the BUT recognizer makes extensive use of discriminatively trained feedforward artificial neural nets to model HMM state posterior probabilities.

We have constructed an English phone recognizer using the BUT architecture from automatically generated STT transcripts on the Switchboard 2 Cell corpora [8]. Training data consisted of approximately 10 hours of speech. ANN training was accomplished using the ICSI Quicknet package [9]. The resulting system has 49 monophones including silence.

The BUT recognizer is used along with the HTK HMM toolkit [10] to produce lattices. Lattices encode multiple hypotheses with acoustic likelihoods. We use the lattice to produce expected counts of tokens and token N-grams.

Expected counts of N-grams can be easily understood as an extension of standard counts. Suppose we have one hypothesized string of tokens, $W = w_1, \cdots, w_n$. Then the 2-grams are created by grouping 2 tokens at a time to form, $W_2 = w_1\_w_2 w_2\_w_3 \cdots w_{n-1}\_w_n$. Higher order N-grams are formed from longer juxtaposition of tokens. The count function for a given bigram, d_i, is $\mathrm{count}(d_i|W_2)$ is the number of occurrences of d_i in the sequence W_2. To extend counts to a lattice, $\mathcal{L}$, we find the ex-

*This work was sponsored by the Department of Defense under Air Force Contract FA8721-05-C-0002. Opinions, interpretations, conclusions, and recommendations are those of the authors and are not necessarily endorsed by the United States Government.

pected count over all all possible hypotheses in the lattice,

$$E_W[\mathrm{count}(d_i|W)] = \sum_{W \in \mathcal{L}} p(W|\mathcal{L})\, \mathrm{count}(d_i|W). \tag{1}$$

A useful extension to expected counts is to find the probability of an N-gram in a lattice. For a single hypothesis, W, we can construct a joint probability using counts of N-grams,

$$\begin{gathered} p(\hat{w}_i, w_i|W) = \frac{\mathrm{count}(\hat{w}_i, w_i|W)}{\sum_j \mathrm{count}(\hat{w}_j, w_j|W)} \\ \hat{w}_i = w_{i-(n-1)}, \ldots, w_{i-1} \end{gathered} \tag{2}$$

where the sum in (2) is performed over all *unique* N-grams in the utterance, and $\mathrm{count}(\hat{w}_i, w_i|W)$ is the number of times the N-gram, $\hat{w}_i w_i$, occurs in the sequence W. We have introduced the notation, $\hat{w}_i$, to denote history (or context) of w_i in the sequence W. Extending (2) to lattices is trivial since we can replace counts by expected counts.

2.2. Discriminative Language Modeling: SVMs

Token-based class recognition using SVMs can be performed in several ways [3, 11]. We focus on an approach which is similar to [11, 2]. In [2], a token sequence, $W = w_1, \cdots, w_N$, is modeled using a bag-of-N-grams. For a sequence of tokens, (joint) probabilities of the unique N-grams, $\hat{w}_j w_j$, on a per conversation basis are calculated, $p(\hat{w}_j, w_j|W)$. Then, the probabilities are mapped to a sparse vector with entries

$$D_j p(\hat{w}_j, w_j|W). \tag{3}$$

The selection of the weighting, D_j, in (3) is critical for good performance. A typical choice is something of the form

$$D_j = \min\left(C_j, g_j\left(\frac{1}{p(\hat{w}_j, w_j|\mathrm{all})}\right)\right) \tag{4}$$

where $g_j(\cdot)$ is a function which squashes the dynamic range, and C_j is a constant. The probability $p(\hat{w}_j, w_j|\mathrm{all})$ in (4) is calculated from the observed probability across all classes. We chose $g_j(x) = \sqrt{x}$ and $C_j = \infty$.

The general weighting of probabilities is then combined to form a kernel between two token sequences, see [2] for more details. For two token sequences, W and V, the kernel is

$$K(W, V) = \sum_j D_j^2 p(\hat{w}_j, w_j|W) p(\hat{w}_j, w_j|V). \tag{5}$$

Intuitively, the kernel in (5) says that if the same N-grams are present in two sequences and the normalized frequencies are similar there will be a high degree of similarity (a large inner product). If N-grams are not present, then this will reduce similarity since one of the probabilities in (5) will be zero. The normalization D_j insures that N-grams with large probabilities do not dominate the kernel function. The kernel can alternatively be viewed as a linearization of the log-likelihood ratio [2].

Incorporating the kernel (5) into an SVM system is straightforward. SVM training and scoring require only a method of kernel evaluation between two objects that produces positive definite kernel matrices (the Mercer condition). We use the package SVMTorch [12]. Training is performed with a one-versus-all strategy. For each target class, we group all remaining class data and then train with these two classes.

3. DISCRIMINATIVE KEYWORD SELECTION

3.1. SVM Feature Selection

A first step towards an algorithm for automatic keyword generation using phones is to examine feature selection methods. Ideally, we would like to select over all possible N-grams, where N is varying, the most discriminative sequences for determining a property of a speech segment. The number of features in this case is prohibitive, since it grows exponentially with N. Therefore, we have to consider alternate methods.

As a first step, we examine feature selection for fixed N and look for keywords with N or less phones. Suppose that we have a set of candidate keywords. Since we are already using an SVM, a natural algorithm for discriminative feature selection in this case is to use a wrapper method [13].

Suppose that the optimized SVM solution is

$$f(X) = \sum_i \alpha_i K(X, X_i) + c \tag{6}$$

and

$$\mathbf{w} = \sum_i \alpha_i \mathbf{b}(X_i) \tag{7}$$

where $\mathbf{b}(X_i)$ is the vector of weighted N-gram probabilities in (3). We note that the kernel given in (5) is linear. Also, the N-gram probabilities have been normalized in (3) by their probability across the entire data set. Intuitively, because of this normalization and since $f(X) = \mathbf{w}^t\mathbf{b}(X) + c$, large magnitude entries in $\mathbf{w}$ correspond to significant features.

A confirmation of this intuitive idea is the algorithm of Guyon, et. al. [14]. Guyon proposes an iterative wrapper method for feature selection for SVMs which has these basic steps:

- For a set of features, $\mathcal{S}$, find the SVM solution with model $\mathbf{w}$.
- Rank the features by their corresponding model entries w_i^2. Here, w_i is the ith entry of $\mathbf{w}$ in (7).
- Eliminate low ranking features.

The algorithm may be iterated multiple times.

Guyon's algorithm for feature selection can be used for picking significant N-grams as keywords. We can create a kernel which is the sum of kernels as in (5) up to the desired N. We then train an SVM and rank N-grams according to the magnitude of the entries in the SVM model vector, $\mathbf{w}$.

As an example, we have looked at this feature selection method for a language recognition task with trigrams (to be described in Section 4). As a motivation for the applicability of Guyon's feature selection method, see Figure 1. The figure shows two functions. First, the CDF of the SVM model values, $|w_i|$, is shown. The figure shows an S-curve shape. There is a relatively large set of small model weights, and a small set of large model weights.

The second curve shows the equal error rate (EER) of the task as a function of applying one iteration of the Guyon algorithm and retraining the SVM. EER is defined as the value where the miss and false alarm rates are equal. All features with $|w_i|$ below the value on the x-axis are discarded in the first iteration. From the figure, we see that only a small fraction ($< 5\%$) of the features are needed to obtain good error rates. This interesting result provides motivation that a small subset of keywords are significant to the task.

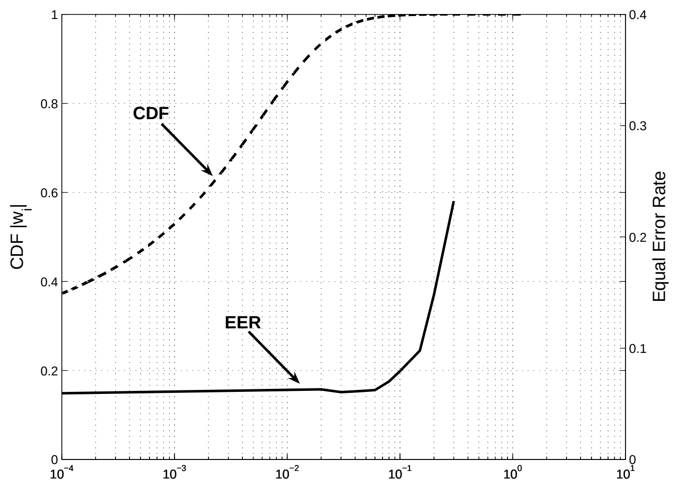

Fig. 1. Feature selection for a trigram language recognition task using Guyon's method

3.2. Keywords via an alternating wrapper/filter method

The algorithm in Section 3.1 gives a method for N-gram selection for fixed N. Now, suppose we want to find keywords for arbitrary N. One possible hypothesis for keyword selection is that since higher order N-grams are discriminative, lower order N-grams in the keywords will also be discriminative. Therefore, it makes sense to find distinguishing lower order N-grams and then construct longer units from these. On the basis of this idea, we propose the following algorithm for keyword construction:

Keyword Building Algorithm

- Start with an initial value of $N = N_s$. Initialize the set, $\mathcal{S}'_N$, to all possible N-grams of phones for this N. For example, $\mathcal{S}_3$ would be the set of all possible trigrams (this is a relatively small set for a phone decoder).
- **Wrapper Step** Apply the feature selection algorithm in Section 3.1 to produce a subset of distinguishing N-grams, $\mathcal{S}_N \subset \mathcal{S}'_N$.
- **Filter Step** Construct a new set of $(N+1)$-grams by juxtaposing elements from $\mathcal{S}_N$ with phones. Nominally, we take this step to be juxtaposition on the left and right, $\mathcal{S}'_{n+1} = \{pd, dq | d \in \mathcal{S}_n, p \in \mathcal{S}_1, q \in \mathcal{S}_1\}$.
- Set $N \leftarrow N+1$, iterate the wrapper and filter steps to some desired order N
- Output: $\mathcal{S}_N$

A few things should be noted about the proposed keyword building algorithm. First, we call the second feature selection process a filter step, since induction has not been applied to produce the $(N+1)$-gram features. Second, note that the purpose of the filter step is to provide a candidate set of possible $(N+1)$-grams which can then be more systematically reduced. Third, several potential algorithms exist for the filter step. In our experiments and in the algorithm description, we nominally append one phone to the beginning and end of an N-gram. Another possibility is to try to combine overlapping N-grams. For instance, suppose the keyword is `some_people` which has the phone transcript `s_ah_m_p_iy_p_l`. Then, if we are looking at 4-grams, we might see as top features `s_ah_m_p` and `p_iy_p_l` and combine these to produce a new keyword.

3.3. Keyword Implementation

Expected N-gram counts were computed on a lattice using the Forward-Backward algorithm. Equation (8) gives the posterior probability of a connected sequence of arcs in the lattice where *src_nd*(a) and *dst_nd*(a) are the source and destination node of arc a, $\ell(a)$is the likelihood associated with arc a, $\alpha(n)$ and $\beta(n)$ are the forward and backward probabilities of reaching node n from the beginning or end of the lattice $\mathbf{L}$ respectively, and $\ell(\mathbf{L})$ is the total likelihood of the lattice (the $\alpha(\cdot)$ of the final node or $\beta(\cdot)$ of the initial node of the lattice).

$$p(a_j, ..., a_{j+N}) = \frac{\alpha(\mathit{src\_nd}(a_j))\ell(a_j)...\ell(a_{j+N})\beta(\mathit{dst\_nd}(a_{j+N}))}{\ell(\mathbf{L})} \tag{8}$$

Now if we define the posterior probability of a node $p(n)$ as $p(n) = \alpha(n)\beta(n)/\ell(\mathbf{L})$. Then equation (8) becomes:

$$p(a_j, ..., a_{j+N}) = \frac{p(a_j)...p(a_{j+N})}{p(\mathit{src\_nd}(a_{j+1}))...p(\mathit{src\_nd}(a_{j+N}))} \tag{9}$$

Which is the product of the posteriors of the arcs along a path divided by the product of the posteriors of the nodes connecting these arcs. Equation (9) is attractive because it provides a way of computing the path posteriors locally using only the individual arc and node posteriors. We use this computation along with a *trie* structure [15] to compute the posteriors of our keywords.

4. EXPERIMENTS

4.1. Experimental setup

The phone recognizer described in Section 2.1 was used to generate lattices across a train and an evaluation data set. The training data set consists of more than 360 hours of telephone speech spanning 13 different languages and coming from a variety of different sources including Callhome, Callfriend and Fisher. The evaluation data set is the NIST 2005 Language Recognition Evaluation data consisting of 2421 utterances for each of the 30 and 10 second conditions. We evaluated our system for both the 30 and 10 second tasks under the NIST 2005 closed condition which limits the evaluation data to 7 languages (English, Hindi, Japanese, Korean, Mandarin, Spanish and Tamil) coming only from the OHSU data source. These are the same conditions as the primary evaluation task for the NIST 2005 evaluation.

The training and evaluation data was segmented using an automatic speech activity detector and segments smaller than 0.5 seconds were thrown out. We also sub-segmented long audio files in the training data to keep the duration of each utterance to around 5 minutes (a shorter duration would have created too many training utterances). Lattice arcs with posterior probabilities lower than 10^{-6} were removed and lattice expected counts smaller than 10^{-3} were ignored. For the wrapper step described in Section 3.2, the top and bottom 600 ranking keywords for each target language in the training data were selected according to the weights in each model. After the filter step, this resulted in a total of 689,516 keywords for the 4-gram, and a total of 831,551 keywords for the 5-gram. The support vector machine was trained using a dual formulation which requires pre-computing all of the kernel inner products between the data points and using an alternate kernel which simply indexes into the resulting kernel matrix (this approach becomes difficult when the number of data points is too large).

phones	keyword
SIL_Y_UW_N_OW	you know
!NULL_SIL_Y_EH_AX	yeah
L_AY_K	like
P_IY_P_AX_L	people
HH_AE_V_AX_N	having
SIL_AY_G_EH_S	I guess
SIL_AY_HH_AE_V	I have
R_IY_L_IY	really
TH_IH_NX_K	think
TH_IH_NX	thing
JH_AH_S_T	just

Table 1. Top ranking keywords for 5-gram SVM

N	1	2	3	4	5
10sec	25.4	16.5	11.3	10.0	13.6
30sec	18.2	07.3	04.3	03.9	05.6

Table 2. English phone recognizer %EER for 10 and 30 second NIST tasks

4.2. Results (Qualitative and Quantitative)

To get a sense of how well our feature selection approach was working, we looked at the top ranking keywords from the English model only (since our phone recognizer is trained using the English phone set). Table 1 summarizes a few of the more compelling phone N-grams (for 3, 4 and 5 grams), and a possible keyword that it corresponds to each one. Not suprisingly, we also noticed that in the list of top-ranking key words there were many variations and partial N-gram matches to what appeared to be the same keyword.

The equal error rate for our system on the NIST 2005 language recognition evaluation is summarized in Table 2. The 4-gram system gave a relative improvement of 20% on the 10 second task and 12% on the 30 second task, but despite the compelling keywords produced by the 5-gram system, the performance actually degraded significantly compared to the trigram and 4-gram systems. A comparison of using BUTs Hungarian tokenizer to our English tokenizer is presented in Table 3 along with results using a standard trigram language model (a standard 4-gram language model was not feasable for us to run). We've also demonstrated a significant reduction in EER by linearly fusing the English and Hungarian SVM scores for both the trigram and the 4-gram. A comparable fusion system using the standard trigram language model fused with a backend trained on held-out dev data does not perform nearly as well.

5. CONCLUSIONS AND FUTURE WORK

We presented a method for automatic construction of keywords given a discriminative speech classification task. Our method was based upon an successively building longer span keywords from shorter span keywords using phones as a fundamental unit. The problem was cast as a feature selection method and an alternating filter and wrapper method was proposed. Results showed that reasonable keywords and improved performance could be achieved using this methodology.

System	Order	30sec	10sec
EN SVM	3	4.3	11.3
EN SVM	4	3.9	10.0
HU SVM	3	5.0	12.2
HU SVM	4	4.6	11.4
Fused EN and HU SVM	3	3.1	8.9
Fused EN and HU SVM	4	2.9	6.5
EN LM	3	5.3	10.9
HU LM	3	4.4	10.0
Fused EN and HU LM	3	4.1	8.9

Table 3. Hungarian and English SVM linear fusion compared to standard LM with backend fusion %EER for 10 and 30 second NIST tasks

6. REFERENCES

[1] A. Stolcke, L. Ferrer, S. Kajarekar, E. Shriberg, and A. Venkataraman, "MLLR transforms as features in speaker recognition," in *Proc. Interspeech*, 2005, pp. 2425–2428.

[2] W. M. Campbell, J. P. Campbell, D. A. Reynolds, D. A. Jones, and T. R. Leek, "Phonetic speaker recognition with support vector machines," in *Advances in Neural Information Processing Systems 16*, Sebastian Thrun, Lawrence Saul, and Bernhard Schölkopf, Eds. MIT Press, Cambridge, MA, 2004.

[3] W. M. Campbell, T. Gleason, J. Navratil, D. Reynolds, W. Shen, E. Singer, and P. Torres-Carrasquillo, "Advanced language recognition using cepstra and phonotactics: MITLL system performance on the NIST 2005 language recognition evaluation," in *Proc. IEEE Odyssey*, 2006.

[4] T. Joachims, *Learning to Classify Text Using Support Vector Machines*, Kluwer Academic Publishers, 2002.

[5] W. M. Campbell, F. Richardson, and D. A. Reynolds, "Language recognition with word lattices and support vector machines," in *Proceedings of ICASSP*, 2007, pp. IV–989 – IV–992.

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

[7] Hynek Hermansky and Sangita Sharma, "Temporal patterns (traps) in asr of noisy speech," in *Proceedings of the International Conference on Acoustics Speech and Signal Processing*, 1999, vol. 1, pp. 289–292.

[8] Linguistic Data Consortium, "Switchboard-2 corpora," http://www.ldc.upenn.edu.

[9] "ICSI QuickNet," http://www.icsi.berkeley.edu/Speech/qn.html.

[10] S. Young, Gunnar Evermann, Thomas Hain, D. Kershaw, Gareth Moore, J. Odell, D. Ollason, V. Valtchev, and P. Woodland, *The HTK book*, Entropic, Ltd., Cambridge, UK, 2002.

[11] Lu-Feng Zhai, Man hung Siu, Xi Yang, and Herbert Gish, "Discriminatively trained language models using support vector machines for language identification," in *Proc. IEEE Odyssey: The Speaker and Language Recognition Workshop*, 2006.

[12] Ronan Collobert and Samy Bengio, "SVMTorch: Support vector machines for large-scale regression problems," *Journal of Machine Learning Research*, vol. 1, pp. 143–160, 2001.

[13] Avrim L. Blum and Pat Langley, "Selection of relevant features and examples in machine learning," *Artificial Intelligence*, vol. 97, no. 1-2, pp. 245–271, Dec. 1997.

[14] I. Guyon, J. Weston, S. Barnhill, and V. Vapnik, "Gene selection for cancer classification using support vector machines," *Machine Learning*, vol. 46, no. 1-3, pp. 389–422, 2002.

[15] Konrad Rieck and Pavel Laskov, "Language models for detection of unknown attacks in network traffic," *Journal of Computer Virology*, vol. 2, no. 4, pp. 243–256, 2007.