LANGUAGE RECOGNITION WITH WORD LATTICES AND SUPPORT VECTOR MACHINES*

W. M. Campbell, F. Richardson, D. A. Reynolds

MIT Lincoln Laboratory
Lexington, MA 02420
E-mail: {wcampbell,frichard,dar}@ll.mit.edu

ABSTRACT

Language recognition is typically performed with methods that exploit phonotactics—a phone recognition language modeling (PRLM) system. A PRLM system converts speech to a lattice of phones and then scores a language model. A standard extension to this scheme is to use multiple parallel phone recognizers (PPRLM). In this paper, we modify this approach in two distinct ways. First, we replace the phone tokenizer by a powerful speech-to-text system. Second, we use a discriminative support vector machine for language modeling. Our goals are twofold. First, we explore the ability of a *single* speech-to-text system to distinguish multiple languages. Second, we fuse the new system with an SVM PRLM system to see if it complements current approaches. Experiments on the 2005 NIST language recognition corpus show the new word system accomplishes these goals and has significant potential for language recognition.

Index Terms— speech processing, natural languages

1. INTRODUCTION

An enduring method for language recognition has been the parallel phone recognition language modeling (PPRLM) system [1]. This system performs recognition with multiple phone decoders that have a null grammar. From the resulting phone sequences, language models are applied for a set of target languages. Ideally, the phone decoder languages should contain the target languages, but even if this is not the case, PPRLM performs well.

A novel extension to the PPRLM system was to add more decoder information via lattices [2]. Lattices provide additional information which can be weighted by the posterior of alternate hypotheses. Since PPRLM relies on a null grammar, the phone acoustic models determine this posterior probability. Extensions to the standard PPRLM perplexity-based scoring with this posterior information are straightforward [2].

In analogy with the high-level feature paradigm in speaker recognition [3], other token sources should be considered for language recognition. In some applications, a word recognizer may already be in use for speech processing and could be leveraged for language recognition. Also, in high-level speaker recognition, word sequences were found to characterize speakers and provide significant complementary information [4]. Word tokens, in a certain sense, model long context phone n-grams, and analyzing this structure should be complementary to the phone-based PPRLM approach.

Traditionally, word systems have been used for language recognition in the following manner [5]. For a set of target languages, speech-to-text (STT) systems in all of these languages are assumed to be available. For a speech input, the STT systems are all applied to produce log-likelihood scores per target language. These scores are combined, by a log-likelihood ratio combination or fusion, to produce posterior probabilities of the target languages which are then used for language recognition. Although this STT language recognition approach is very accurate [5], the computational complexity is large. In addition, STT systems must be available in the target languages of interest—a challenging task.

As a novel alternative, we propose using the lattice output of a *single* STT decoder. This new configuration has substantially lower complexity than the standard approach and requires less language resources to build. The lattice output is used in a discriminative support vector machine (SVM) for language recognition. SVMs can effectively work with sparse data and include standard PRLM scoring as a special case.

The outline of the paper is as follows. In section 2, we discuss language recognition using lattices. Section 3 describes the application of SVMs to language recognition. Finally, in section 4, we present a series of experiments that uses the new techniques on the NIST language recognition evaluation (LRE) corpus.

2. LANGUAGE RECOGNITION USING LATTICES

Suppose we have an STT decoder, language models $p(\cdot|\cdot, L)$ for target languages L, and the most-likely hypothesis (a sequence of words) produced by the STT decoder, W^*. The standard PRLM approach [1] is to find the language that maximizes the log likelihood,

$$\begin{aligned} L^* &= \operatorname*{argmax}_L \log p(W^*|L) \\ &= \operatorname*{argmax}_L \frac{1}{N} \sum_{i=1}^{N} \log p(w_i|w_{i-(n-1)}, \cdots, w_{i-1}, L) \end{aligned} \tag{1}$$

where N is the number of words in the sequence W^*, n is the n-gram order, and $W^* = w_1, \cdots, w_N$.

An alternate expression of (1) can also be used. We can construct a joint probability from W^* using counts of n-grams,

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

[*] 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.

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^*. Then, (1) can be expressed using (2) and the negative log of the perplexity, s_L,

$$\begin{aligned} L^* &= \operatorname*{argmax}_L s_L(W^*) \\ s_L(W^*) &= \sum_i p(\hat{w}_i, w_i|W^*) \log p(w_i|\hat{w}_i, L). \end{aligned} \tag{3}$$

Note that in the equation, $p(\hat{w}_i, w_i|W^*)$ is a joint probability, and the sum is over all unique n-grams in W^*. The quantity s_L in (3) is maximized when the target language model has the same distribution as the language model from the utterance, see [6]. Intuitively, from an information theory perspective, (3) measures the (negative) expected bit length of coding a source with distribution $p(\cdot|W^*)$ using an assignment of $\log p(\cdot|\cdot, L)$ bits.

Standard PRLM scoring (3) was extended to lattices by Gauvain, et. al. [2], by taking the expectation of s_L over the set of hypotheses $\mathcal{W}$ in the lattice. Performing this operation gives

$$E_{\mathcal{W}}[s_L(W)] = \sum_{\hat{w}w} E_{\mathcal{W}}[p(\hat{w}, w|W)] \log p(w|\hat{w}, L) \tag{4}$$

where in this case the sum is performed over the unique n-grams, $\hat{w}w$, in the lattice. An alternate expression for (4) is

$$E_{\mathcal{W}}[s_L(W)] = \sum_{W \in \mathcal{W}} p(W|X, \Lambda, \Phi) s_L(W) \tag{5}$$

where Λ is the STT acoustic model, Φ is the STT language model, and X are the acoustic observations. The first formulation is useful since we can produce expected counts of different n-grams across a lattice using a forward-backward algorithm, apply (2), and then insert the resulting probabilities in (4). The second formulation (5) is interesting since it explicitly shows the weighting of different log perplexity scores by the posterior of the hypothesis and their dependence on the acoustic and language model of the STT decoder.

A question in interpreting (5) is how the posterior, $p(W|X, \Lambda, \Phi)$, should be computed for an STT system. In the general formulation, the STT system would produce an acoustic model score, $p(X|W, \Lambda)$ for the input X, and a language model score, $p(W|\Phi)$. These would be weighted and combined to produce a score

$$\alpha \log(p(X|W, \Lambda)) + \beta \log(p(W|\Phi)) \tag{6}$$

Computing the posterior $p(W|X, \Lambda, \Phi)$ is straightforward, and we introduce another factor, γ, to obtain

$$p(W|X, \Lambda, \Phi) = \frac{p(X|W, \Lambda)^{\alpha/\gamma} p(W|\Phi)^{\beta/\gamma}}{\sum_{W' \in \mathcal{W}} p(X|W', \Lambda)^{\alpha/\gamma} p(W'|\Phi)^{\beta/\gamma}}. \tag{7}$$

The role of γ in (7) is to vary the distribution of the posterior over W. As $\gamma \to \infty$, $p(W|X, \Lambda, \Phi)$ will approach a uniform distribution; as $\gamma \to 0$ the distribution will approach a Kronecker delta with mass at the most likely path. The factor γ is commonly used in discriminative training algorithms such as MMIE [7].

The role of (α, β, γ) in language recognition is difficult to determine. For example, we may want to use an α and β different from the STT decode (in the extreme case $\beta = 0$) to emphasize the acoustic model more and the STT language model less. We may also want to increase γ to increase the contribution of non-optimal paths in the computation of expected counts. We explore these issues experimentally in Section 4.

3. SUPPORT VECTOR MACHINES FOR TOKEN-BASED LANGUAGE RECOGNITION

A support vector machine, $f(x)$, is a two-class classifier formed from sums of a kernel function, $k(\cdot, \cdot)$,

$$f(x) = \sum_i \alpha_i k(x, x_i) + d. \tag{8}$$

The weights $\alpha_i > 0$ and d are trained using an optimization algorithm, and the x_i, called support vectors, are selected from a labeled training data set. The kernel is expressible as inner product, $k(x, y) = b(x)^t b(y)$, where $b(\cdot)$ is usually in a high dimensional space. This inner product form also allows the SVM to be written as $f(x) = w^t b(x) + d$.

Token-based language recognition using SVMs can be performed in several ways [8, 9, 10]. We focus on an approach which is similar to [8, 10, 11]. In [11], a token sequence, $W = w_1, \cdots, w_N$, is modeled using a bag-of-n-grams approach. For a sequence of tokens, (joint) probabilities of the unique n-grams, $\hat{w}_j w_j$, on a per conversation basis are calculated, $p(\hat{w}_j, w_j|W)$. As an example, for words, a typical bigram might be $\hat{w}_2 w_2 = \texttt{the\_example}$. Then, the probabilities are mapped to a sparse vector with entries

$$D_j p(\hat{w}_j, w_j|W). \tag{9}$$

The selection of the weighting, D_j, in (9) is critical for good performance. A typical choice is something of the form

$$D_j = \min\left(C_j, g_j\left(\frac{1}{p(\hat{w}_j, w_j|\mathrm{all})}\right)\right) \tag{10}$$

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 (10) is calculated from the observed probability across all classes. The squashing function should monotonically map the interval $[1, \infty)$ to itself to supress large inverse probabilities. Typical choices for g_j are $g_j(x) = \sqrt{x}$ and $g_j(x) = \log(x) + 1$. In both cases, the squashing function g_j normalizes out the typicality of a feature across all classes. The constant C_j limits the effect of any one feature on the kernel inner product. If we set $C_j = 1$, then this makes $D_j = 1$ for all j.

The general weighting of probabilities is then combined to form a kernel between two token sequences, see [12] for more details. For two token sequences, W and V, the kernel is

$$K(W, V) = \sum_j D_j^2 p(\hat{w}_j, w_j|W) p(\hat{w}_j, w_j|V). \tag{11}$$

Intuitively, the kernel in (11) 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 (11) 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 [12].

Incorporating the kernel (11) 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 [13]. Training is performed with a one-versus-all strategy. For each target language, we group all remaining language data into another class and then train with these two classes.

Extending (11) to lattices is straightforward. Rather than computing the probability $p(\hat{w}_j, w_j|W)$ using counts from the 1-best sequence, expected counts from a lattice can be used.

The SVM resulting from the kernel in (11) has similarities with the log perplexity scoring (3) described in Section 2. If we simplify the SVM by explicitly expanding the kernel using (8) and (11), we obtain

$$f(W) = \sum_j m_j p(\hat{w}_j, w_j|W) + d \tag{12}$$

where $m_j = \sum_i \alpha_i D_j^2 p(\hat{w}_j, w_j|W_i)$, and the W_i are the support vectors. We can see that if $m_j = \log p(w_j|\hat{w}_j, L)$ then the SVM model matches the standard log perplexity scoring [10].

Several additional items should be noted about the kernel in (11). First, the choice of squashing function and C_j interacts with the token type in many ways. For tokens like phones, it is not uncommon to see all unigrams and bigrams in a conversation; $g_j(x) = \sqrt{x}$ and $C_j = \infty$ works well in this case. For tokens that are infrequently repeated such as words, there may be many small expected counts which causes small probabilities in both the background and a given conversation. We have found that a more aggressive squashing function, $g_j(x) = \log(x) + 1$ and a $C_j = \infty$ as an example, limit the possibility of one n-gram dominating the inner product. Of course, an alternate approach would be to prune out low-frequency features—essentially a feature selection process. Secondly, for the kernel (11), different n-gram types can be combined by summing kernels for different n. That is, for example, we can calculate a new kernel using a sum of a unigram kernel and a bigram kernel.

4. EXPERIMENTS

4.1. Experimental Setup

For our experiments, we used the NIST LRE 2005 data set and the 30 second task. We split this data set into two parts by random selection resulting in two data sets of approximately 1800 utterances. One of these data sets was used for training language recognition models and the other half was used for testing. This setup was used to minimize the amount of STT decoding and as a proof of concept.

For language recognition, we used the BBN Byblos English STT system [14] to produce lattices. SVM models were trained using the kernel given in (11) with squashing function $\log(x) + 1$ and $C_j = \infty$. For the SVM, the kernel used both unigram and bigram probabilities. SVM models for the target languages for the NIST evaluation task were produced using SVMTorch using $c = 10$ for regularization.

We did not use any backend fusion process for scoring as in [8]. Instead, we computed likelihood ratios across languages assuming that the SVM score can be treated approximately as a log likelihood ratio [8]. Although this impacts performance, it allows us to avoid decoding and scoring on a separate development set.

4.2. Decoding Experiments

We experimented with multiple decoding strategies to determine the best approach for language recognition. Segmentation for decoding was performed using a GMM-based speech activity detection system. Byblos was applied to these segments and output lattices were considered after two major passes. The first pass considered, UDEC, produced lattices from an unadapted decode of the input segments. The other pass considered, ADEC, was the final decode from the system using various forms of adaptation. A 1-best path was derived from the lattices of the UDEC pass.

We examined several 1-best outputs of the STT system to see how the decoder responded to non-English inputs. Interestingly, the system responded with nonsensical outputs containing a large variety of vocabulary and structure. As an example, for a section of the Mandarin utterance *lid00020*, the following output was produced:

> ... Um ... Which_I don’t_know_if I_can I_could_be tempting though ...

Anecdotally, the output produced by the recognizer could be “understood” while listening to the speech.

We then performed recognition experiments using the various decoding passes. Performance was measured using the equal error rate (EER). We initially considered only an English/non-English detection task (English), but then expanded to the complete 7 target language task in the NIST LRE (7L) with no out-of-class inputs. Expected counts were computed from the lattices using standard Byblos acoustic model and language model weights and a γ, see (7), that made the language model exponent β/γ approximately $1/5$. We will comment more on this choice in Section 4.3. Results are summarized in Table 1.

From Table 1, we can see several properties of the system. First, not only is the system able to do English/non-English detection, it is also able to differentiate non-English languages based on STT output. This implies the STT system is producing systematically structured lattices even for atypical input data. Second, lattice-based methods outperform a 1-best approach; this intuitively should be the case since a lattice has more information content. Third, the UDEC pass outperforms the ADEC pass. The likely source of this degradation is that the ADEC pass is performed using the UDEC lattice as a constraint. The combination of this constraint, sharper adapted acoustic models, and pruning combine to a create a lattice that is not as complex in the ADEC pass.

4.3. Varying posterior parameters

We next considered varying the parameters for the exponents shown in (7). Results are shown in Table 2. From the table, we see that a large γ value improves accuracy. This result implies that the different hypothesis in the lattice should be weighted more equally when computing expected counts. Another observation is that the language model only $(\alpha, \beta) = (0, 1)$ and acoustic model only $(\alpha, \beta) = (1/12, 0)$ results are similar. This may be because pruning of the large search space is biased by a large weight on the language model. Overall, these parameters are not highly sensitive, except that γ must be large.

4.4. Comparisons and Fusion

We compared the SVM word system to multiple systems and also fused the results. As a baseline, we considered the log-likelihood

Table 1. Comparison of EER for different decoding passes using an SVM language recognition system with word lattices

Task	Decode Pass	EER (%)
English	1-best	14.2
English	UDEC	5.6
English	ADEC	9.8
7L	1-best	14.2
7L	UDEC	11.5
7L	ADEC	18.4

Table 2. Comparison of EER for different weightings for posterior computation

α	β	γ	EER Eng (%)	EER 7L (%)
1/6	1	1	7.4	14.1
1/12	1	1	6.5	13.6
1/12	1	2	5.6	11.9
1/12	1	5	5.6	11.5
1/12	1	1000	5.6	11.4
1/12	0	5	5.8	11.4
0	1	5	5.8	11.7

Table 3. Comparison of EER for different LID systems and fusion

System	EER Eng (%)	EER 7L (%)
UDEC STT log likelihood	30.7	-
ADEC STT log likelihood	24.8	-
Phone SVM	5.7	8.2
Word SVM	5.6	11.4
Fuse SVM phone & word	3.2	6.2

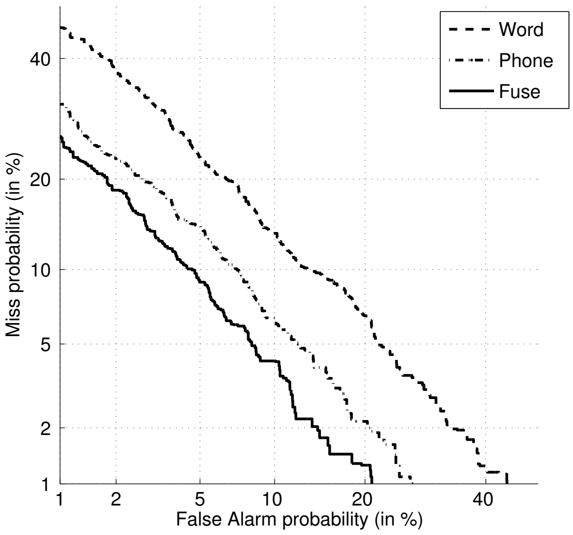

Fig. 1. Plot of the SVM word system, the SVM phone system, and linear fusion

per frame output by the recognizer as an English/non-English detector. We also included an SVM phone system based upon 6 OGI-TS trained tokenizers, see [8]. The SVM phone system was trained on the same data set as the SVM word system. Finally, we fused the SVM phone system with the SVM word system (fusion) using an equally weighted linear combination of SVM outputs.

Results are shown in Table 3 and Figure 1. From the table, we see that using the *raw* log-likelihood score from the STT system is not a good English/non-English detector. Interestingly, the ADEC pass performance is better than the UDEC pass which contrasts to the opposite performance we obtained with the SVM system in Section 4.2. The SVM phone system performs better than the SVM word system on the 7L task—this might be expected since the SVM phone system has smaller acoustic units. Finally, note that the fusion of the SVM word and SVM phone system performs significantly better showing the potential of the new approach in conjunction with standard approaches.

5. CONCLUSIONS

We have successfully demonstrated the use of SVMs and word lattices for language recognition. The SVM word system performed well on a standard NIST LRE data set. We addressed several key issues necessary for tuning performance—lattice generation and posterior calculation. Potential future work includes comparing the approach to standard language modeling methods, applying the methods on a larger data set, and using advanced fusion methods.

6. REFERENCES

[1] M. Zissman, "Comparison of four approaches to automatic language identification of telephone speech," *IEEE Trans. Speech and Audio Processing*, vol. 4, no. 1, pp. 31–44, 1996.

[2] J. L. Gauvain, A. Messaoudi, and H. Schewenk, "Language recognition using phone lattices," in *Proceedings of ICSLP*, 2004.

[3] Douglas Reynolds, Walter Andrews, Joseph Campbell, Jiri Navratil, Barbara Peskin, Andre Adami, Qin Jin, David Klusacek, Joy Abramson, Radu Mihaescu, Jack Godfrey, Doug Jones, and Bing Xiang, "The SuperSID project: Exploiting high-level information for high-accuracy speaker recognition," in *Proceedings of ICASSP*, 2003, pp. IV–784–IV–787.

[4] J. P. Campbell, D. A. Reynolds, and R. B. Dunn, "Fusing high- and low-level features for speaker recognition," in *Proc. Eurospeech*, 2003, pp. 2665–2668.

[5] J. Navratil, "Automatic language identification," in *Multilingual Speech Processing*, T. Schultz and K. Kirchhoff, Eds., pp. 233–272. Academic Press, 2006.

[6] F. Jelenik, *Statistical Methods for Speech Recognition*, MIT Press, 1997.

[7] P. C. Woodland and D. Povey, "Large-scale MMIE training for conversational telephone speech recognition," in *Proc. NIST Transcription Workshop*, 2000.

[8] William Campbell, Terry Gleason, Jiri Navratil, Douglas Reynolds, Wade Shen, Elliot Singer, and Pedro Torres-Carrasquillo, "Advanced language recognition using cepstra and phonotactics: MITLL system performance on the NIST 2005 language recognition evaluation," in *Proc. IEEE Odyssey: The Speaker and Language Recognition Workshop*, 2006.

[9] Christopher White, Izhak Shafran, and Jean-Luc Gauvain, "Discriminative classifiers for language recognition," in *Proceedings of ICASSP*, 2006, pp. I–213–I–216.

[10] 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.

[11] William M. Campbell, Joseph P. Campbell, Douglas A. Reynolds, Douglas A. Jones, and Timothy 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.

[12] W. M. Campbell, J. P. Campbell, D. A. Reynolds, D. A. Jones, and T. R. Leek, "High-level speaker verification with support vector machines," in *Proceedings of ICASSP*, 2004, pp. I–73–76.

[13] 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.

[14] S. Matsoukas, R. Prasad, B. Xiang, L. Nguyen, and R. Schwartz, "The RT04 BBN 1xRT recognition systems for English CTS and BN," in *Proceedings of the Rich Transcription Workshop*, 2004.