IMPROVED PHONOTACTIC LANGUAGE IDENTIFICATION USING RANDOM FOREST LANGUAGE MODELS

XiaoRui Wang[1], ShiJin Wang[1], JiaEn Liang[1], Bo Xu[1,2]

[1]Digital Media Content Technology Research Center,
[2]National Lab of Pattern Recognition,
Institute of Automation, Chinese Academy of Sciences, Beijing, China, 100080

ABSTRACT

Recently a new language model, the random forest language model (RFLM), has been proposed and shown encouraging results in speech recognition tasks. In this paper we applied the RFLM to language identification tasks. We proposed a shared backoff smoothing to deal with data sparseness problem. Experiments were conducted on a subset of NIST 2003 language recognition evaluation data. The RFLM obtained 15.7% relative error rate reduction comparing with the standard trigram LM. The RFLM can be used as a counterpart to n-gram LM and BTLM for system fusion. We also empirically studied the relation between system performance and the tree numbers in a RFLM.

Index Terms— Random Forest Language Models, Language Identification, Decision Tree Language Models

1. INTRODUCTION

One of the most efficient approaches for language identification (LID) is parallel phone recognition followed by language modeling (PPRLM) [1]. This approach is based on the assumption that phonotactic constraints contain enough information to identify the languages. The phone decoders used in PPRLM systems can be limited to a few languages and the language models can be trained as long as unlabeled data are available. This makes the PPRLM systems very effective both in terms of decoding time and needed resources. The PPRLM systems are widely used and perform consistently well[1, 2, 3].

The most popular language model used in PPRLM systems is n-gram, which makes the following approximation:

$$P(w_i|w_1, w_2, \cdots, w_{i-1}) \approx P(w_i|w_{i-n+1}^{i-1}) \qquad (1)$$

where w_i^j denotes the tokens $w_i \cdots w_j$, which is also called history. The model order is often 2 or 3. The n-gram model often suffers from data sparseness problem and may give unsatisfied results. Smoothing techniques, such as model interpolation, are used to deal with this problem.

The work is supported by National High Technology Research & Development program 863 of China under contract 2006AA010103.

In addition to smoothing techniques, there are also researches focusing on data sharing or clustering. One of these is the decision tree language model (DTLM). The DTLM clusters similar histories into equivalence classes and each history in a class shares the same distribution over the predicted tokens. In [4] a binary decision tree language model (BTLM) was successfully applied to LID tasks and has shown favorable results comparing to standard n-grams. It is also used as an effective counterpart of n-gram LMs in fusion. One problem of DTLMs is that the decision tree construction procedure also suffers from the data sparseness problem because the tree splitting algorithms decide only on seen data. Another problem is that the node splitting algorithm is greedy and may not lead to the optimal decision tree.

Recently a random forest language model (RFLM) [5] is proposed and successfully used in speech recognition systems [5, 6]. The RFLM is a natural extension to DTLMs. A RFLM is a collection of randomly constructed DTLMs and the model probability is the average of the probabilities from all the DTLMs. In this paper we applied RFLMs to PPRLM systems. We also proposed a shared backoff smoothing algorithm for RFLMs. Experiments were conducted on a subset of NIST 2003 language recognition evaluation data. We also compared the results of RFLMs with n-gram LMs and BTLMs.

The rest of the paper is organized as follows. In section 2, we review the decision tree language models. In section 3, we describe the RFLM and our shared backoff smoothing algorithm. In section 4 we give our experimental results and analysis. Finally in section 5 we draw some conclusions.

2. DECISION TREE LANGUAGE MODELS

In decision tree language models (DTLM), n-grams are classified into equivalence classes by asking questions about tokens in a specific history position. Each node has a history position i and two disjoint set L and R for asking questions. For an n-gram, let w_i denote its token at position i. If $w_i \in L$, it will proceed to the left child node. If $w_i \in R$, it will proceed to the right child node. If w_i are not in L and R, it can

not proceed further.

Following [5] our decision tree training procedure consists two stages: a growing stage and a pruning stage.

2.1. Growing a Decision Tree

The growing stage starts from a single node, the root node, which is also the only leaf node. The root node contains all n-grams of the training data. Leaf nodes are split into new nodes recursively by a sequence of node splitting operations. After splitting a node, n-grams in the node are also split into new sub nodes. We select log-likelihood as our stop criterion and grow the decision tree as deep as possible, until the log-likelihood is maximized.

Considering to split a node N using two disjoint sets L and R at history position i, the log-likelihood of the training data will be

$$\mathcal{L}(D|L,R) = \sum_w [(C(w,L)\log\frac{C(w,L)}{C(L)} + C(w,R)\log\frac{C(w,R)}{C(R)}] \quad (2)$$

where $C(w,\cdot)$ is the frequency count of token w following all histories in $(\cdot)$ and $C(\cdot)$ is the corresponding total count. For each history position i, we will choose two best subsets, $L*$ and $R*$, which give the maximum log-likelihood. We use a greedy exchange algorithm to find $L*$ and $R*$ as follows:

1. Let L contains all tokens of position i in the n-grams of current nodes. Let R be empty.
2. For each token w_l in L, if move w_l to R leads to increase log-likelihood then add it to R and remove it from L.
3. For each token w_r in R, if move w_r to L leads to increase log-likelihood then add it to L and remove it from R.
4. If any exchange was made in step 2 or step 3, return to step 2.

We examine each position in the history and choose the best one which can increase the log-likelihood the most. If for all history positions the log-likelihood cannot be increased by any L and R, we will mark the node as leaf node and stop splitting from it. The whole growing procedure can be summarized as follows:

1. Let N denote the leaf node under consideration for splitting. Initially N is the root node.
2. For each position $i(i = 1, \cdots, n-1)$ in the history, find two best subsets L_i* and R_i* as described earlier.
3. Choose a best position $i*$ from the $n-1$ positions in step 2 which increase the log-likelihood the most.

$$i* = \arg\max_{1\le i\le n-1} \mathcal{L}(D|L_i*, R_i*) \quad (3)$$

Use $i*$, L_i* and R_i* to split N, mark N as internal node and mark the new nodes as leaf nodes.

4. Repeat step 2 and 3 until no node splitting can increase the log-likelihood.

2.2. Pruning a Decision Tree

The pruning stage aims to avoid over training and to get a more robust tree structure. Heldout data is used to prune the decision tree. We also use the log-likelihood as the pruning criterion. If the log-likelihood on the heldout data cannot be increased by growing a node into sub-trees, we will prune the sub-trees rooted in that node.

The decision tree is fully built after pruning and all the leaf nodes are used as equivalence classes to estimate token probabilities. Note that the DTLM is different from the BTLM in [4]. For a node in the BTLM, its question is associated with a history position i and a token set S. If w_i is in S the n-gram will go to the left child, otherwise it will go to the right child. So any n-gram can reach a leaf node in the BTLM. While in the DTLM an n-gram may stop at an internal node. For such n-grams we simply use their lower order probabilities.

3. RANDOM FOREST LANGUAGE MODELS

A random forest language model is a collection of randomized DTLMs, whose trees may be viewed as i.i.d. samples from a subset of possible decision trees. From section 2.1 we can see that for DTLMs the tree growing algorithm is greedy. It is optimal locally and may not lead to the optimal decision tree. For RFLMs, the decision tree growing algorithm is randomized. A node splitting may not be locally optimal. The decision trees in a RFLM randomly classify the training data into different equivalence classes and may capture different characteristics of the data. After combining all the trees in the forest, the RFLM can get better results than greedily grown DTLMs.

3.1. Randomizing a Decision Tree

In [5] a randomized decision tree growing algorithm was proposed and we briefly review it here.

There are three ways to construct a randomized decision tree. The first is to randomly select history positions. The second is to randomly initialize the two subsets, L and R. The third is to randomly sample the training data. We use the first two methods because training data sampling makes the data more sparse.

To randomly select a history position, for each of the $n-1$ positions we have a Bernoulli trial with a probability r for success. The $n-1$ trials are assumed to be independent of each other. The positions that have successful trials are chosen as a subset. For each position in the subset, we try to split the node with randomly initialized sets L and R using the greedy exchange algorithm described in section 2.1. We then chose the position that has the best split.

3.2. Shared Backoff

The training data in a leaf node may still sparse, because even for large training data unseen contexts are also possible. Most DTLMs use bottom-up recursive smoothing [4] to deal with this problem. The probability can be calculated as

$$P_{bu}(w|N) = \alpha P(w|N) + (1-\alpha)P(w|parent(N)) \quad (4)$$

where $P_{bu}(w|N)$ is the bottom-up smoothed probability in node N, $P(w|N)$ is the token w's probability in node N, α is a weight parameter and $parent(N)$ is the parent node of N. We found this method doesn't work well for RFLMs. Although it may help a single DTLM, the bottom-up recursive smoothing can reduce the randomness of the trees in a forest and makes all the DTLMs toward a same distribution. Here we propose a shared backoff algorithm. For unseen data, we combine a lower order model with a backoff weight. All histories in a leaf node share the same backoff weight. Let $C_N(w)$ be the total count of tokens w dropped in the node N that

$$C_N(w) = \sum_{h \in N} C(hw) \quad (5)$$

where $C(hw)$ is the frequency count of the sequence hw. Let S be the set of lower order histories that

$$S(N) = \{h' | h \in N\} \quad (6)$$

where h' is the history obtained by dropping the first token in h. The probability after shared backoff smoothing can be defined as follows:

$$P_{sbo}(w|h) = \begin{cases} P(w|N) & if\ C_N(w) > 0 \\ \alpha(N) \sum_{h' \in S(N)} P(w|h') & if\ C_N(w) = 0 \end{cases}$$

where $\alpha(N)$ is the shared backoff weight and can be calculated to make the total probabilities sum to one.

$$\alpha(N) = \frac{1 - \sum_{C_N(w)>0} P(w|N)}{\sum_{w_i \in V, C_N(w_i)=0} \sum_{h' \in S(N)} P(w_i|h')} \quad (7)$$

Because there are no model structure constraints for the lower order model, it can be any LMs. In this paper we use n-gram LM as the lower order model.

4. EXPERIMENTS

Our experiments were conducted on a subset of NIST 2003 language recognition evaluation data. Six languages were chosen: Egyptian Arabic, American English, German, Korean, Mandarin Chinese and Spanish. We choose these six languages because we don't have the LDC CallFriend data. We use the CallHome and LDC2003S03(Korean) data as our training data instead. We don't use Japanese evaluation data since it is from the CallHome corpus[7]. We use NIST LID Dev'96 and Eval'96 data sets as our development set and use Eval'03 data as test set. We test on the 30s duration set.

4.1. System Description

Two phone recognizers trained on English and Mandarin separately were used in our experiments. The acoustic feature vectors are 39-components comprised of 12 MFCC cepstrum coefficients and the log energy, along with their first and second order derivatives. The acoustic models are context independent phone models. Each phone model is a tied-state HMM with Gaussian mixture observation densities and 16 Gaussians per state. The acoustic models are trained on our training corpora described above. We use 44 phones for English and 39 phones for Mandarin. The text corpus used to train language models are phone transcriptions of training data, decoded by the two phone recognizers.

Outputs of the two recognizers were fused using Gaussian backend classifiers. The raw language model scores were stacked into feature vectors, which are transformed by applying linear discriminant analysis and used for training the Gaussian classifiers. We estimate the Gaussian distribution of the respective score vectors for every language, so we will have a Gaussian distribution for each language in the system. Our LID system framework is shown in figure 1.

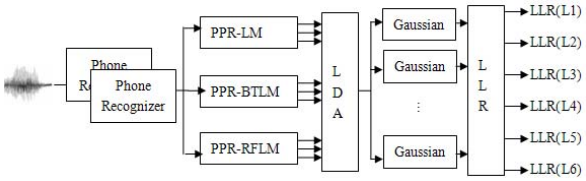

Fig. 1. *System framework*

4.2. Language Identification Results

In the first experiment we first compared the LID results by using n-gram LM, BTLM and RFLM and then fused these three results using the backend.

For the baseline PPRLM system, smoothed back-off trigram LMs were used. 6 language dependent trigram models were trained on each phone transcript. For comparison we also built BTLMs as described in [4]. Leaf adaptation and bottom-up smoothing were also used and the model order is 3.

To build RFLMs, the training data transcripts were used to growing the decision trees and Dev'96 and Eval'96 data was used as heldout data to prune the trees. Peng Xu's RFLM tool [8] was used to train the model. For each language, 20 decision trees were randomly built to form a forest. After the forest was constructed, our shared backoff was used to smooth the LM and use a language dependent bigram as the lower order model. Since there are a number of trees, the recognition speed may become slow because we have to search every tree to get a token's probability. We pre-calculated all the trigram's probabilities from a RFLM before test and stored the

probabilities in a table. When scoring only table lookup operations were needed. The RFLM's order is also set to 3. The error rate of LID results are shown in table 1.

LM	Error Rate(%)	Improv.(%)
Trigram	10.94	-
BTLM	10.16	7.1
RFLM	9.22	15.7
fusion	8.75	20.0

Table 1. *Language identification results using Trigram LM, BTLM, RFLM and system fusion.*

As we can see from table 1, the RFLM achieves considerable improvements. The BTLM obtained 7.1% improvements comparing with the standard trigram LM and the RFLM obtained 15.7% improvements. There are two basic reasons. Firstly by data clustering the model's complexity is reduced and the model parameters can be estimated more robustly. Secondly by randomly building the trees, each tree in the forest captured different characteristics of the training data and combining all the trees will help to improve the performance.

We then fused the three results using the backend and got an error rate of 8.75%, 20% relative improvements from the baseline. This result shows that the RFLM can be used as a counterpart of BTLMs and n-gram LMs in fusion.

4.3. Selecting Tree Numbers

In the second experiment we empirically studied how the tree numbers can affect the RFLM's performance. We used different tree numbers to train several RFLMs and test for each number on our LID system. The numbers we test are 1, 2, 3, 4, 5, 10, 15, 20, 25 and 30. Because the tree growing algorithm is randomized, to get reliable results we run 5 times for each number and use the average error rate. The results are shown in figure 2. Results of n-gram LM and BTLM were also plot in the figure for comparison.

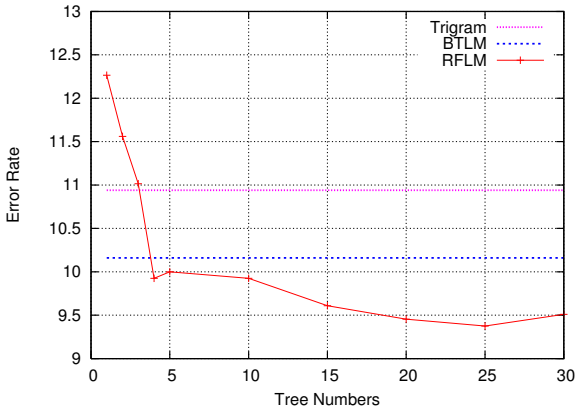

Fig. 2. *Error rate as a function of tree numbers*

Figure 2 shows the dependency of error rate on the number of trees used in RFLM. We can see that when the tree number is 3 the RFLM get similar results to the n-gram LM. Using only 4 trees the RFLM outperforms the n-gram LM and the BTLM. After that the error rate slightly goes down when increasing the tree number. It shows that 20 or 25 are adequate tree numbers.

5. CONCLUSIONS

In this paper we successfully applied the RFLM to language identification tasks. A shared backoff smoothing technique was proposed to deal with data sparseness problem. Experiments showed encouraging results. The RFLM obtained 15.7% relative error rate reduction comparing with a standard n-gram baseline. The PPR-RFLM system can be used as a counterpart to n-gram and BTLM systems for system fusion. We empirically studied the relation between system performance and the tree numbers in a RFLM and found that 20 or 25 are adequate numbers.

6. REFERENCES

[1] M.A. Zissman, "Comparison of four approaches to automatic language identification of telephone speech," *IEEE Transanctions on Speech and Audio Processing*, vol. 4(1), pp. 31–44, January, 1996.

[2] P. Matjka, P. Schwarz, J. ernock, and P. Chytil, "Phonotactic language identification using high quality phoneme recognition," in *Interspeech'2005*, 2005, pp. 2237–2240.

[3] W. Shen, W. Campbell, T. Gleason, D. Reynolds, and E. Singer, "Experiments with lattice-based pprlm language identification," in *Speaker and Language Recognition Workshop, 2006.*, 2006.

[4] J. Navratil, "Recent advances in phonotactic language recognition using binary-decision trees," in *Proceedings of Interspeech-2006*, 2006.

[5] P. Xu and F. Jelinek, "Random forests in language modeling," in *Proceedings of EMNLP'2004*, 2004.

[6] Y. Su, F. Jelinek, and S. Khudanpur, "Large-scale random forest language models for speech recognition," in *Proceedings of Interspeech-2007*, 2007.

[7] A. F. Martin and M. A. Przybocki, "Nist 2003 language recognition evaluation," in *Proceedings of Interspeech-2003*, 2003.

[8] URL, "http://www.clsp.jhu.edu/people/xp/," .