EURASIP Journal on Audio, Speech, and Music Processing

SpringerOpen

Phone lattice reconstruction for embedded language recognition in LVCSR

EURASIP Journal on Audio, Speech, and Music Processing 2012,
2012:15 doi:10.1186/1687-4722-2012-15

Yuxiang Shan (shanyuxiang@gmail.com)
Yan Deng (y-deng05@mails.tsinghua.edu.cn)
Jia Liu (liuj@tsinghua.edu.cn)
Michael T Johnson (michael.johnson@marquette.edu)

ISSN 1687-4722

Article type Research

Submission date 15 July 2011

Acceptance date 13 April 2012

Publication date 13 April 2012

Article URL http://asmp.eurasipjournals.com/content/2012/1/15

Phone lattice reconstruction for embedded language recognition in LVCSR

Yuxiang Shan*[1], Yan Deng[1], Jia Liu[1], and Michael T Johnson[1,2]

[1]Tsinghua National Laboratory for Information Science and Technology, Department of Electronic Engineering, Tsinghua University, Beijing 100084, China

[2]Department of Electrical Engineering, Marquette University, Milwaukee, 53201 WI, USA

*Corresponding author: shanyuxiang@gmail.com

Email addresses:

YD: y-deng05@mails.tsinghua.edu.cn

JL: liuj@tsinghua.edu.cn

MTJ: michael.johnson@marquette.edu

Abstract

An increasing number of multilingual applications require language recognition (LRE) as a frontend, but desire low additional computational cost. This article demonstrates a novel architecture for embedding phone based language recognition into a large vocabulary continuous

speech recognition (LVCSR) decoder by sharing the same decoding process but generating separate lattices. To compensate for the prior bias introduced by the pronunciation dictionary and the language model of the LVCSR decoder, three different phone lattice reconstruction algorithms are proposed. The underlying goals of these algorithms are to override pronunciation and grammar restrictions to provide richer phonetic information. All of the new algorithms incorporate a vector space modeling backend for improved LRE accuracy. Evaluated on a Mandarin/English detection task, the proposed integrated LVCSR-LRE system using frame-expanded N-best phone lattice achieves comparable performance to a state-of-the-art phone recognition-vector space modeling (PRVSM) system, but with an added computational cost three times lower than that of a separate PRVSM system.

Keywords: language recognition; speech recognition; lattice reconstruction; vector space modeling.

---

1 Introduction

Applications such as speech-to-speech translation systems and dialogue systems often work in a multilingual environment, so it is necessary to rapidly identify the language being spoken. Even for monolingual systems, language recognition (LRE) is necessary for out-of-language (OOL) detection [1, 2] to filter out segments of non-target languages. One common approach to implement language recognition is to make use of the automatic speech recognizers (ASRs) which already exist in the system. The user's utterance can be decoded by a recognizer for each supported language, with the language type determined by the recognizer that returns the highest ASR score. This approach typically obtains high

accuracy for LRE [3], but also has two obvious disadvantages in that it works only with multiple ASR frontends and it is not suitable for real-time applications, since the LRE decision can be made only after all the decoding processes complete. If there are more than two or three supported languages, the high computational cost can make this approach infeasible. Another approach is to build a dedicated language identifier which runs separately before any other processing steps take place. This approach requires less computational power, but causes a delay in the response of the system. A third approach is to perform LRE in parallel with speech recognition in an assumed core language, restarting the recognition process if the initial hypothesized language is incorrect [4]. The parallel architecture has faster average response time with delayed response only when the initial hypothesized language is incorrect. However, one deficiency of this architecture is that the LRE and the ASR process are independent of each other, leading to additional computational cost. By using an ASR based LRE methods, the computational cost can be significantly reduced by tightly integrating them together. Lattice-based LRE techniques [5–8] which model the context of high level features such as words or phones are effective methods that could be suitable candidates for implementing this approach.
In this article, we introduce a method to improve the parallel LRE-ASR architecture by embedding a phone recognition followed by vector space modeling (PRVSM) LRE backend into an LVCSR decoder to reduce total computational cost. This new embedded architecture has substantially lower complexity than the existing ones and thus requires less computational resources. The difficulty of this integration is that the VSM backend cannot directly make use of phone lattices or transcriptions generated by the LVCSR decoder because of the system's integrated pronunciation dictionary and language model (LM). In an LVCSR decoding network, the pronunciation dictionary restricts within word phone connections and the n-gram LM constraints cross-word connections to provide high recognition accuracy. In contrast, a standard PRVSM system uses a simple phone loop and null grammar, which guarantees that each phone has equal probability to be recognized. If the LVCSR lattices were to be used for LRE, it would give a heavy bias towards the initial target language and cause great performance degradation. To overcome this bias and improve LRE accuracy, we propose to use a separate lattice reconstruction algorithm using

the internal back-tracking information collected during LVCSR decoding. In this article, three different lattice reconstruction algorithms are evaluated, and their LRE accuracies are tested and compared.

There are, of course, other alternatives to VSM for the LRE backend, such as an LM approach [5]. In this study, we choose VSM because of its superior performance [9, 10]. There are also other LRE techniques, such as those that use long term spectral features like shifted delta cepstrum [11], which do not share the same underlying acoustic features or decoding structure with LVCSR, and would not be as well suited for direct integration as the phone recognition approach. By using the same core recognition engine, the new method proposed here allows for strong LRE accuracy accompanied by significantly reduced computational cost.

This article is outlined as follows. Section "Embedding LRE into LVCSR" introduces the proposed embedded LRE architecture, while Section "LVCSR decoding and phone lattice generation" outlines the LVCSR decoding and phone lattice generation, and Section "Phone lattice reconstruction algorithms" details the three phone lattice reconstruction algorithms. Section "Vector space modeling LRE backend" gives a brief description of the specific VSM LRE backend used in this article. Section "Experimental results" presents experimental results and some discussions.

2 Embedding LRE into LVCSR

Figure 1 illustrates the proposed embedded LRE architecture and a typical work flow for a bilingual application. In this framework, language L1 is initially predicted. As the user begins to speak, his speech is sent to L1's decoder. During L1's decoding progress, phone back-tracking information is collected and used to generate phone lattices, which are used by the VSM backend to identify the language type. If the recognized language is the same as L1, the L1's decoding progress is continued as is the L1's post processing, including tasks such as translation and keyword spotting. However, if the language recognition classifies the speech as a different language, for example L2, the current decoding progress is terminated and L2's recognition decoder is activated. The final output would then be

generated by L2's processing chain.

The proposed architecture differs from [4] in three aspects. First, the language recognition is no longer a separate process, but uses the existing LVCSR decoder as its lattice generation frontend, saving computational time. Second, in [4] only phone sequences are used for language recognition, whereas in the proposed architecture, phone lattices are used. Lattices provide significant additional information about non-optimal decoding paths, which may be especially useful when the recognition accuracy is low. Third, a VSM backend instead of a language model is used, which has recently been shown to have superior performance [9].

3 LVCSR decoding and phone lattice generation

In this section, we give mathematical descriptions of each part of the embedded architecture. Given an observation sequence $\mathbf{O}$, the LVCSR decoder decides the most probable word sequence $\mathbf{W}^*$ using Bayes' rule:

$$\mathbf{W}^* = \arg\max_{\mathbf{W}} p(\mathbf{W}|\mathbf{O}) \ = \arg\max_{\mathbf{W}} \left\{ p\left(\mathbf{O}|\mathbf{W}\right) p\left(\mathbf{W}\right) \right\}. \tag{1}$$

With phone based acoustic models, Equation (1) can be further written as:

$$\mathbf{W}^* = \arg\max_{\mathbf{W}} \left\{ p\left(\mathbf{W}\right) \sum_{\mathbf{P}} p\left(\mathbf{O}|\mathbf{P}\right) p\left(\mathbf{P}|\mathbf{W}\right) \right\}, \tag{2}$$

where $\mathbf{P}$ is a phone sequence corresponding to word sequence $\mathbf{W}$. During decoding, if we record each word hypotheses along with its phone sequence, we can obtain phone-level recognition results. This is the most common method for phone lattice generation in LVCSR, and the generated phone lattices can be viewed as exact phone alignments of the word lattice. For clarity, we refer to this conventional phone hypothesis approach as "word-based phone alignment", and the generated phone lattices as "word-based phone lattices".

In Equation (2), $p(\mathbf{W})$ is the language model constraining inter-word connections, and $p(\mathbf{W}|\mathbf{P})$ is the pronunciation dictionary restricting phone connections within each word. In most cases, the relation between $\mathbf{P}$ and $\mathbf{W}$ is a one-to-one mapping. As will be shown in

Section "Experimental results", the word-based phone lattices obey these constraints, resulting in a heavy bias towards the target language and causing significant LRE performance degradation. To remove these constraints and improve LRE accuracy, we propose to use an alternative construction algorithm for phone lattice generation.

4 Phone lattice reconstruction algorithms

The phone lattice reconstruction algorithms improve LRE performance by removing pronunciation and grammar constraints to provide a richer set of decoding alternatives and context. A "bag of phones" phone hypothesis approach is used to achieve this goal.
As opposed to "word-based phone alignment", the "bag of phones" approach collects phone hypotheses generated at each frame during LVCSR decoding into a phone bag regardless of the word it belongs to. If a phone hypothesis of the same identity and start/end frame number already exists, for implementation simplicity only the one with a larger likelihood score is kept. Once a phone hypothesis is put into the bag, it remains, even if the path that it belongs to is pruned. In this way, we can identify a large number of prospective phone hypotheses that never appear in the word-based phone lattices. To avoid data sparseness in the VSM LRE backend, left and right triphone contexts are removed and only the underlying monophones are preserved in the phone bag. These collected phone hypotheses are then used to construct new phone lattices from scratch.
Another reason for ignoring phone context-dependency constraints is that this allows the generated lattices to contain richer monophone hypotheses. In the following algorithms, an N-best phone list is constructed at each frame from the "bag-of-phones" by choosing phone hypotheses with the top-N largest likelihood scores. Considering storage and computational resource limitations, we cannot choose a very large N ($N = 10$ in our experiments). Since context-dependent variants of the same phone are acoustically similar and thus have correlated likelihoods, allowing multiple context-dependent variants in the N-best list has the effect of limiting phonetic diversity, particularly for smaller values of N.
To maintain clear notation, we define a lattice $L = (N, A, n_{\text{start}}, n_{\text{end}})$ as a directed acyclic graph (DAG) which is specified by a finite set of nodes N, a finite set of links (or arcs) A, a

start node $n_{\text{start}} \in N$ and an end node $n_{\text{end}} \in N$. Given an arbitrary node $n \in N$, $T[n]$ denotes its occurrence time identified by frame number, and $N[t] \in N$ is the inverse function returning the corresponding lattice node at time t. Given a link $a \in A$, $S[a]$ denotes its start node, $E[a]$ its end node, $I[a]$ its hypothesis identity, $ac[a]$ its acoustic likelihood, $lm[a]$ its language model likelihood and $pr[a]$ its posterior probability. $|N|$ represents the total number of nodes, and $|A|$ represents the number of links. For a phone hypothesis p, $S[p]$ denotes its start frame number, $E[p]$ its end frame number, $I[p]$ its phone identity and $ac[p]$ its acoustic likelihood.

4.1 Time-aligned phone lattice

In the first lattice construction algorithm, we apply a time alignment algorithm to the word-based phone lattices to remove the dictionary and LM constraints. This method is called "time-aligned" because it clusters the start and end time nodes of each lattice link, and thus aligns all the links topologically into a linear (sausage-like) structure, where each alignment corresponds to an equivalence class of phone hypotheses, and the ordering of the equivalence classes is consistent with that of the original lattice. The phone connections in the word-based phone lattice are broken, and the phone arcs are reorganized by their partial ordering relations. Therefore, the pronunciation and grammar constraints no longer exist. This approach reuses phone hypotheses that already exist in the word-based phone lattices, and does not change the conventional LVCSR decoding procedure. The proposed algorithm is similar to the lattice alignment algorithm used in building a confusion network [12]. A heuristic clustering procedure is used to group time overlapped links into clusters based on lattice topology, time and phonetic hypotheses, and transform the lattice into a linear graph where all paths pass through all nodes. The precedence order of the links in the original lattice is preserved. Although some time information is lost, the aligned phone lattice can be used for language recognition, because the phone connection patterns characterized by their n-gram statistics are much more important. The primary difference between the proposed algorithm and the confusion network algorithm is that we do not measure phonetic similarity between phones, but only cluster based on time. Similarly to the confusion network approach, we first define an equivalence class and

partial ordering. Given a link $a \in A$, $[a]$ denotes its equivalence class of aligned phone hypotheses. Given a lattice, we can define a partial ordering $\leq$ on its links. The ordering is defined by the following: for $a, b \in A$, $a \leq b$ iff $a = b$ or $E[a] = S[b]$ or $\exists c \in A$ such that $a \leq c$ and $c \leq b$. Put more simply, $a \leq b$ means a comes before b. For two equivalence classes $[a]$ and $[b]$, the partial ordering $[a] \preceq [b]$ implies $a_1 \leq b_1$, $\forall a_1 \in [a], b_1 \in [b]$. Given a phone lattice, the basic alignment procedure is that of finding an ordered link equivalence that is consistent with the lattice ordering and is also a total ordering. The algorithm is summarized as follows:

Step 1. Decode the input speech utterance with an LVCSR decoder, collecting phone back-tracking information using word-based phone alignment to generate a word-based phone lattice. Left and right triphone contexts in the generated lattice are removed.
Step 2. Initialize link equivalence classes by phone identity and start and end times t_1, t_2:

$$C_{p,t_1,t_2} = \{a : I[a] = p, S[a] = t_1, E[a] = t_2, a \in A\}. \tag{3}$$

Let $X = \{C_{p,t_1,t_2} : \text{for all } p, t_1 \text{and} t_2\}$ be the set of initial equivalence classes.
Step 3. From X, choose the two most similar unordered equivalence classes C_1^* and C_2^*, and merge them into a new equivalence class C_{new}:

$$(C_1^*, C_2^*) = \underset{\substack{C_1 \in X, C_2 \in X \\ C_1 \not\preceq C_2, C_2 \not\preceq C_1}}{\arg\max} \text{SIM}(C_1, C_2). \tag{4}$$

The similarity between two equivalence classes C_1 and C_2 is measured by:

$$\text{SIM}(C_1, C_2) = \max_{\substack{a_1 \in C_1 \\ a_2 \in C_2}} \text{overlap}(a_1, a_2) \cdot pr[a_1] \cdot pr[a_2], \tag{5}$$

where $\text{overlap}(a_1, a_2)$ is defined as the time overlap between the two links normalized by the sum of their lengths.
Step 4. $\forall C \in X$, update partial ordering relations: Set $C \preceq C_{\text{new}}$, if $C \preceq C_1^*$ or $C \preceq C_2^*$; set $C_{\text{new}} \preceq C$, if $C_1^* \preceq C$ or $C_2^* \preceq C$.
Step 5. $\forall (C_1, C_2) \in X \times X$, update partial ordering relations: Set $C_1 \preceq C_2$, if $C_1 \preceq C_1^*$ and $C_2^* \preceq C_2$, or $C_1 \preceq C_2^*$ and $C_1^* \preceq C_2$.
Step 6. Set $X = X \cup \{C_{\text{new}}\} \backslash \{C_1^*, C_2^*\}$.

Step 7. Repeat Steps 3 to 7 until there are no unordered equivalence classes left.
Step 8, Output the converted lattice.

This algorithm has a time complexity $O\left(|A|^3\right)$. Figure 2 gives an example of a lattice constructed using this algorithm. For readability, node and link scores are not marked. As shown, the phone hypotheses are aligned topologically, with pronunciation constraints removed and duplicated paths from different LM states merged. A side effect of the algorithm is that some phones which originally formed a sequence, for example "IAO3", "M", and "AI4" in Figure 2b, end up as a parallel set. Our experimental study showed that this usually happens when (1) the durations of the paralleled phones are very short. For example, the durations of "M" and "AI4" are 0.03 s and 0.04 s, respectively, which are much shorter than other phones in the lattice; (2) the time intervals of the paralleled phones have significant overlap with other phones. For example, "M" and "AI4" are completely overlapped with "IA2" and "IA4"; (3) the paralleled phones are often mis-recognized, with very low posterior probabilities. In the example of Figure 2, the log posterior probabilities of "IAO3", "M" and "AI4" are -72.57 (not marked in the figure), which have little impact on the expected n-gram count and negligibly impact language recognition performance.

4.2 Phoneme-expanded 1-best lattice

The key idea of the time-aligned phone lattice approach is to find the best time alignment in order to remove pronunciation dictionary and LM constraints. In the second algorithm, we further enhance this approach by incorporating richer phonetic information. To reduce the complexity, we first use the 1-best phone transcription generated by the LVCSR decoder as a reference alignment to divide the whole speech utterance into time slots, and then fill each slot using N-best phone hypotheses. Connecting the time slots one by one, we can obtain an expanded phone lattice. The algorithm is summarized as follows:

Step 1. Decode the input speech utterance with an LVCSR decoder, collecting phone back-tracking information in both the word-based phone alignment and bag-of-phone methods. Generate an initial phone-level transcription of the speech utterance and a bag of phone hypotheses.

Step 2. From the initial phone transcription, get a list of phone boundaries $B = \{t_0, t_1, \ldots, t_{M-1}\}$, where M is the number of boundaries.
Step 3. Initialize phone lattice $L = (N, A, n_{\text{start}}, n_{\text{end}})$ with $N = \{N[t] : \forall t \in B\}$, $A = \{\}$, $n_{\text{start}} = N[t_0]$ and $n_{\text{end}} = N[t_{M-1}]$.
Step 4. For $i = 1$ to $M - 1$, find the N-best phone hypotheses with start time t_{i-1} and end time t_i, and store them into list P. For each phone hypothesis p in P, add a new link a to A with $S[a] = N[t_{i-1}]$, $E[a] = N[t_i]$, $I[a] = I[p]$, $ac[a] = ac[p]$, $lm[a] = 0$.
Step 5. Output L.

Figure 3 gives an example lattice constructed using this algorithm, using the same utterance as in Figure 2. Figure 3a is the initial phone transcription, while Figure 3b is the new phone lattice constructed from the N-best phone hypotheses. Although lattices reconstructed by this algorithm have a similar topology to the time-aligned lattice, the scores attached to each node and link have different meanings, with scores identified as likelihoods in this algorithm and posterior probabilities in the time-aligned lattice. Another major difference is that the start/end times in the phoneme-expanded 1-best lattice are correct, whereas they are an adjusted approximation in the time-aligned lattice.

4.3 Frame-expanded N-best lattice

Both of the above two algorithms use optimal alignments derived from LVCSR results, which may implicitly incorporate pronunciation and grammar constraints. In the third algorithm, we try to completely eliminate the constraints of the pronunciation dictionary and language model, and reconstruct a new phone lattice from scratch. This is achieved by concatenating the N-best phone hypotheses of each frame. In contrast to the phoneme-expanded case, the N-best order is decided by duration normalized log likelihood:

$$\text{score}_{\text{Norm}}(p) = \frac{\log ac[p]}{E[p] - S[p]} \tag{6}$$

where p is a phone hypothesis. The algorithm is summarized as follows:

Step 1. Decode the input speech utterance with an LVCSR decoder, collecting phone back-tracking information using the bag of phones approach. Use this to generate a bag of phone hypotheses.

Step 2. Initialize phone lattice $L = (N, A, n_{\text{start}}, n_{\text{end}})$ with $N = \{N[t] : t = 0, 1, \ldots, M\}$, $A = \{\}$, $n_{\text{start}} = N[0]$ and $n_{\text{end}} = N[M]$, where M is the number of frames.
Step 3. For $i = 1$ to M, find the N-best phone hypotheses with ending frame number i, and store them into list P. For each phone hypothesis p in P, add a new link a to A with $S[a] = N[S[p]]$, $E[a] = N[i]$, $I[a] = I[p]$, $ac[a] = ac[p]$, $lm[a] = 0$.
Step 4. Remove unreachable nodes and links from L.
Step 5. Output L.

Step 4 is necessary for correct n-gram counting. Once a phone hypothesis is pushed into the bag-of-phones, token pruning has no effect on it. As a result, the phone bag contains many phone hypotheses from pruned paths. In this algorithm, phone hypotheses are concatenated frame by frame. In most cases, these pruned paths can be connected to other surviving paths and make their way to the end node. However, there may be occasional truncated paths, which can cause failure in forward and backward n-gram counting.
With no time alignment, lattices reconstructed using this algorithm encode the richest possible phonetic information and are typically quite large, which requires larger temporary storage space. However, after n-gram counting in the VSM backend all the lattices can be deleted. In practice, a lattice is converted to an n-gram super-vector immediately after it is generated, so the storage requirement does not increase significantly. Figure 4 shows an example using the same utterance as in Figures 2 and 3. Since the entire lattice is much too large to include, only the part of the first word "DONG1" is drawn.

4.4 Pruning frame-expanded N-best lattice

The use of N-best lists in the third algorithm causes two deficiencies in the generated lattices: First, due to the continuity of the speech signal and overlapping frames during LVCSR feature extraction, one frame's N-best phone candidates are very likely to still be present in the next frame's N-best phone list. This can be seen clearly from Figure 4, in which N-best phone candidates are duplicated frame by frame. Second, the use of the N-best list can force low-probability phone hypotheses to be recorded, especially during silence regions. These result in additional lattice redundancy, which if not removed may

affect the N-gram statistics, and decrease the discriminability of the classifier. To eliminate some of that redundancy, we introduce a beam pruning mechanism into the N-best phone list generation procedure.

Suppose $P(t)$ denotes a list of phone hypotheses with ending frame number t, we can define the duration normalized score of the best hypothesis at time t as:

$$BS\left(t\right) = \max_{p \in P(t)} \left\{ \text{score}_{\text{Norm}}\left(p\right) \right\}. \tag{7}$$

The pruning criterion then states that those phone hypotheses $p \in P\left(t\right)$ are pruned for which

$$\text{score}_{\text{Norm}}\left(p\right) < BS\left(t\right) - T, \tag{8}$$

where the threshold T determines the width of the beam.

5 Vector space modeling LRE backend

To do language recognition, we implement a VSM backend similar to [9]. Figure 5 shows the complete framework of phone recognition followed by vector space modeling. In a standard PRVSM system, a phone recognizer with a null grammar is used to tokenize input utterances to phone lattices and corresponding n-gram statistics. In our proposed embedded architecture, the phone recognizer is replaced by an LVCSR decoder and lattices are generated and reconstructed using the algorithms described above. Denoting $S = s_1, \ldots, s_N$ as an arbitrary phone sequence in the lattice, the expected n-gram counts of the resulting lattice L can be expressed as:

$$\begin{aligned} \text{count}\left(s_i, \hat{s}_i | L\right) &= E_S\left[\text{count}\left(s_i, \hat{s}_i | S\right)\right] \\ &= \sum_{S \in L} p\left(S|L\right) \text{count}\left(s_i, \hat{s}_i | S\right) \end{aligned} \tag{9}$$

where $\hat{s}_i = s_{i-(n-1)}, \ldots, s_{i-1}$ represents the n-gram history and n is the n-gram order. The count function returns the number of occurrence of a given n-gram entry $(s_i, \hat{s}_i)$ in phone sequence S. The posterior probability $p\left(S|L\right)$ can be computed efficiently by the forward-backward algorithm as described in [6]. From the counts, the joint probability of

an n-gram entry in a lattice is calculated as:

$$p\left(s_i, \hat{s}_i|L\right) = \frac{\text{count}\left(s_i, \hat{s}_i|L\right)}{\sum_j \text{count}\left(s_j, \hat{s}_j|L\right)}. \tag{10}$$

For a given lattice, we calculate the joint probabilities for all unique n-grams in the lattice and arrange them into a super-vector, which are then taken as data for SVM (support vector machine) training and testing for language recognitions.

It has been shown that n-gram normalization significantly improves performance [13]. We choose the term frequency log likelihood ratio kernel [13] to weight and normalize the n-gram super-vectors and form a linear kernel for classification. The kernel function between two super-vectors is:

$$K\left(X_1, X_2\right) = \sum_{i=1}^{N} \frac{p\left(s_i, \hat{s}_i|L_1\right)}{\sqrt{p\left(s_i, \hat{s}_i|all\right)}} \cdot \frac{p\left(s_i, \hat{s}_i|L_2\right)}{\sqrt{p\left(s_i, \hat{s}_i|all\right)}}, \tag{11}$$

where X_1 is the super-vector for lattice L_1, X_2 is the super-vector for lattice L_2, $p\left(s_i, \hat{s}_i|all\right)$ is calculated across lattices derived from all the train data. During training, the inputs of this kernel are two arbitrary super-vectors in the training data. For testing, one of the inputs is a super-vector of a test utterance and the other is a SVM support vector.

Figure 6 shows a diagram of a parallel PRVSM (PPRVSM) system, which fuses phonetic features from multiple phone recognizers. N-gram statistics are computed from each recognition lattice using the PRVSM approach and the super-vectors are concatenated to form the input for SVM. PPRVSM has been shown to yield better performance than a single PRVSM system [5].

6 Experimental results

6.1 Experimental setup

To evaluate the proposed architecture and algorithms, we implemented a Mandarin/English language recognition task. Two LVCSR decoders were built as frontends, one for Mandarin and the other for English. For both decoders, PLP with delta and acceleration coefficients were used as features, with cepstral mean/variance

normalization. For the Mandarin frontend, a 68 k-word dictionary was used, and an acoustic model of 6,000 states and 48 mixtures was trained with about 300 hours data selected from the training sets of the HKUST, CallFriend, CallHome, and Chinese 863 corpora. For the English frontend, a 130 k-word dictionary was used, and an acoustic model of 6,000 states and 48 mixtures was trained with about 220 h data selected from the training set of CallFriend, CallHome, and Switchboard. The Mandarin and the English language model were trained with the Chinese and English Gigaword corpora, respectively, and interpolated with transcriptions of the acoustic model training data. Acoustic model training was done using the minimum phone error (MPE) [14] criteria.

For the VSM backend, trigram probability super-vectors were extracted following the steps described in Section "Vector space modeling LRE backend", with a core phone set of 96 monophones for Mandarin and 39 monophones for English, excluding sil and sp. This resulted in super-vectors of length 894,048 for Mandarin and 60,879 for English. The training corpus consisted of 9,475 Mandarin utterances and 10,827 English utterances selected from the CallFriend and CallHome training sets. Each training utterance was automatically segmented to have about 30 s of speech. For testing, we selected 14,467 utterances, representing approximately 7.2 h of English and 5.5 h of Mandarin from the CallHome development and test sets. Each test utterance was segmented according to the LDC provided transcriptions. Utterances shorter than 0.5 s are discarded. Table 1 gives the utterance length distribution of the test data. In all experiments, the training and test utterances were converted into lattices using identical lattice reconstruction algorithms to make training and test conditions match.

For each lattice reconstruction algorithm, a corresponding VSM language recognition system was implemented, as shown in Table 2. In the following experiments, the performance of these systems is examined. SVMTorch [15] is used for SVM training and testing. For comparison, we built a Mandarin phone recognizer and an English phone recognizer, each as part of a standard PRVSM baseline. The PRVSM frontends use a simple phone-loop and null grammar. These share the same features with the LVCSR decoder, but use significantly compacted context-dependent acoustic models which have only about 300 states. The acoustic model used in the PRVSM system is tuned on NIST

LRE 07 data to provide better performance [16]. Alternative English and Mandarin PRVSM frontends using the LVCSR acoustic models are also evaluated, which are named "PRVSM-alt" in the Table 2. In our implementation, the PRVSM-alt frontend is efficiently integrated with the LVCSR decoder by sharing the same preprocessing and acoustic model evaluation. Although both the PRVSM frontend and the PRVSM-alt frontend generate lattices using context-dependent phone sets, the phone contexts are removed prior to the n-gram counting in the VSM backend to avoid data sparseness.

6.2 Performance of the different lattice reconstruction algorithms

Detection error trade-off (DET) curves for the LVCSR Mandarin frontend experiments are shown in Figure 7, with corresponding equal error rates (EERs) given in Table 3. The DET plots using the English frontend have a similar profile and are not presented. From the results, we observe that all three lattice reconstruction algorithms improve the LRE performance significantly compared to direct use of the word-based phone lattice obtained from LVCSR, with the frame-based N-best method A3 outperforming the other two algorithms. Because this approach completely discards topological (LM, lexicon) constraints used in the LVCSR decoder, and resorts to rearranging phone hypotheses in the bag of phones frame by frame, the resulting phone lattices have the richest phonetic information but the fewest pronunciation and grammar constraints. The fact that the PRVSM-alt performs only slightly better than the A3 method supports this constraint-removal efficiency. However, it is interesting to notice that A1 has the advantage of containing lower false alarm and miss rate operating points.

From Table 3, we can see that the Mandarin frontend performs better than the English frontend. This is most likely due to the size of the phone set. In our systems, the English phone set has 39 monophones and the Mandarin phone set has 96 excluding sil and sp. A larger LVCSR phone set means both English and Mandarin utterances can be modeled more precisely in the n-gram vector space, leading to better language recognition performance.

Although both use a phone-loop topology, there is still a performance gap between the PRVSM and the PRVSM-alt frontends. We attribute this to the different characteristics of

the acoustic models for PRVSM language recognition and LVCSR. A PRVSM system is motivated by the observation that some sequences of phones that exist in one language rarely exist in another. In such kind of systems, the phone recognizer tokenizes input speech into phone sequences or lattices, and the VSM backend converts phone sequences or lattices into super-vectors and then makes decision. It is very common to use an English frontend, for example, to tokenize Mandarin or speech of many other languages. Therefore, the frontend's generality and robustness to multiple languages are much more important than precisely transcribing speech of a specific target language which is the purpose of a LVCSR decoder. The acoustic models of the LVCSR decoder use many more states and Gaussian mixtures to distinguish phones in different contexts, which is more accurate for LVCSR but less robust for language recognition, and thus leads to performance degradation.

6.3 Effect of pruning frame-expanded N-best lattice

As explained in Section "Pruning frame-expanded N-best lattice", removing low-probability phone hypotheses from the N-best list can reduce redundancy and confusability of frame-expanded N-best lattices, and improve LRE performance. Figure 8 plots the EER of A3 as a function of percentage of links retained from the original (unpruned) lattice. We observe that with 80–85% of the links in the original lattice we obtain about 0.3% improvement (0.32% for Mandarin and 0.29% for English) in EER over the original lattice. Figure 7 and Table 3 also give DET plot and EERs of the pruned lattices where the 84% beam width is used for Mandarin and 81% for English.
Lattice pruning is not applied to the time-aligned phone lattice or the phoneme-expanded N-best lattice. These two algorithms avoid some of the redundancy and confusability through the incorporation of time alignment, for direct lattice alignment (time node clustering) in algorithm A1 and from the use of the 1-best transcription in algorithm A2. However, this lower confusability comes at the cost of implicit incorporation of language-specific pronunciation dictionary and language model constraints, which as shown in the previous section results in lower LRE accuracy.

6.4 Performance of parallel frontends

As discussed earlier, fusion of multiple phonetic features generally improves performance. To evaluate the impact of this for the new integrated LVCSR-LRE system, we tested the system using parallel Mandarin and English LVCSR frontends. Figure 9 gives the resulting DET plots, with corresponding EERs listed in Table 4. As expected, system performance is significantly improved in all cases. This result demonstrates the importance of having a highly diverse phone set for the language recognition task, and of using parallel frontends if computational resources permit.

6.5 Performance of different utterance length

In this experiment, we examined the impact of utterance length on language recognition accuracy. Figure 10 shows EERs of the parallel frontend system with different test utterance lengths. The general trend is that EER decreases as utterance lengths get longer, with more reliable language recognition results using test segments longer than three seconds. This provides us an intuitive frame of reference as to how long it may take to get a reliable language recognition result, and thus an idea of the expected added LVCSR latency possible in the case of incorrect initial language settings.

6.6 Computational cost

Computational costs for the proposed LVCSR-based lattice construction algorithms vary significantly. Both A1 and A2 require collection of phone hypotheses using word-based phone alignment. Since phone hypotheses are propagated, merged and pruned with word hypotheses, this requires the decoder to spend significant time on bookkeeping. However, this is not needed for the A3 approach, because the bag-of-phones method for phone candidate identification is much simpler. Table 5 lists the specific additional computational cost of the lattice reconstruction algorithms, compared to using a separate PRVSM LRE system. The test was conducted on a single core of an Intel Core2 Duo 2.26G CPU. In the implementation of the PRVSM system, the frontend re-uses features extracted by the LVCSR frontend, but the acoustic model evaluation is not shared because the acoustic

models are different. It should be noted that A3 and A3' have the same real time factor. This is because the most time consuming part of the lattice reconstruction algorithm is the phone hypothesis collection which is Step 1 of the A3 method, with the computational cost of the other part being trivial. Since A3' also has significantly better performance than the other approaches it is likely a better choice for real application.

The PRVSM-alt having a larger computational overhead does not seem consistent with intuition. Since the PRVSM-alt and the LVCSR decoder are tightly integrated by sharing the same preprocessing and acoustic model evaluation, the overall system should have a similar overhead to that of A3. However, we found that the PRVSM-alt frontend has a much larger number of HMM states evaluated on each frame. In our experiments, the PRVSM-alt Mandarin frontend has about 4,800 states/frame on average, while the LVCSR decoder has only about 3,000 states/frame. The acoustic evaluation takes nearly 1xRT in the PRVSM-alt system, but 0.66xRT in the pure LVCSR, beam search in the phone-loop also takes about 0.1xRT, so the overall computational cost increases. This is caused by different search strategies and the decoding network topology. The lexicon and the language model constraints in the LVCSR decoder provides various and powerful pruning criteria (such as word-end pruning, language model look-ahead pruning, etc.) to be applied in the decoding progress to shrink the search space, but only acoustic pruning is available for the phone-loop decoder. In a pronunciation prefix tree based LVCSR decoder, such as the one we used in this article, the acoustic states near word-ends have fewer opportunities to be evaluated because the active search space near the word-ends normally have been extensively pruned. However, in a phone-loop decoder where all the phones are connected in parallel, all the acoustic models have equal probability of evaluation. Of course, a narrower beam can be used for faster decoding speed, but this usually leads to even worse accuracy.

7 Conclusion

In this article, we have successfully demonstrated an embedded LVCSR-LRE architecture that integrates language recognition capability directly into an LVCSR decoder with very

low additional computational cost. The bias introduced by the LVCSR pronunciation dictionary and language model is reduced through the use of a bag-of-phones approach that allows for encoding richer decoding alternatives. Of the three proposed lattice reconstruction algorithms, the integrated LVCSR-LRE system using the frame-expanded N-best phone lattice shows the best performance, which is comparable to a state-of-the-art PRVSM LRE system but at an added computational cost three times lower than that of a separate PRVSM system.

Abbreviations

ASR, automatic speech recognition; DAG, directed acyclic graph; DET, detection error trade-off; EER, equal error rate; LM, language model; LRE, language recognition; LVCSR, large vocabulary continuous speech recognition; MPE, minimum phone error; OOL, out-of-language; PPRVSM, parallel phone recognition - vector space modeling; PRVSM, phone recognition - vector space modeling; SVM, support vector machine; VSM, vector space modeling.

Competing interests

The authors declare that they have no competing interests.

Acknowledgements

This research was supported by the National Natural Science Foundation of China (NSFC) (Nos. 90920302 and 61005019), by the NSFC and Research Grants Council (RGC) of Hong Kong (No. 60931160443), and in part by the National High Technology Development Program of China (863 Program) (No. 2008AA040201).

References

1. P Motlicek, Automatic out-of-language detection based on confidence measures derived from LVCSR word and phone lattices, in *Proc. Interspeech 2009*, Brighton, UK, 2009, Vol. 1, pp. 1215–1218

2. P Motlicek, F Valente, PN Garner, English spoken term detection in multilingual recordings, in *Proc. Interspeech 2010*, Chiba, Japan, 2010, Vol. 1, pp. 206–209

3. F Fernandez, RD Cordoba, J Ferreiros, V Sama, LF D'Haro, Language identification techniques based on full recognition in an air traffic control task, in *Proc. ICSLP 2004*, Jeju, Korea, 2004, Vol. 1, pp. 1565–1568

4. DCY Lim, I Lane, Language identification for speech-to-speech translation, in *Proc. Interspeech 2009*, Brighton, UK, 2009, Vol. 1, pp. 204–207

5. M Zissman, Comparison of four approaches to automatic language identification of telephone speech. IEEE Trans. Speech Audio Process. **4**(1), 31–44 (1996)

6. JL Gauvain, A Messaoudi, H Schewenk, Language recognition using phone lattices, in *Proc. ICSLP 2004*, Jeju, Korea, 2004, Vol. 1, pp. 25–28

7. WM Campbell, E Singer, PA Torres-Carrasquillo, DA Reynolds, Language recognition with support vector machines, in *Proc. Odyssey 2004*, Toledo, Spain, 2004, Vol. 1, pp. 285–288

8. WM Campbell, F Richardson, DA Reynolds, Language recognition with word lattices and support vector machines, in *Proc. ICASSP 2007*, Honolulu, Hawaii, 2007, Vol. 4, pp. 989–992

9. H Li, B Ma, CH Lee, A vector space modeling approach to spoken language identification. IEEE Trans. Audio Speech Lang. Process. **15**(1), 271–284 (2007)

10. Y Deng, J Liu, Automatic language identification using support vector machines and phonetic n-gram, in *Proc. International Conference on Audio, Language and Image Processing (ICALIP) 2008*, Shanghai, China, 2008, Vol. 1, pp. 71–74

11. PA Torres-Carrasquillo, Language identification using Gaussian mixture models. PhD thesis, Michigan State University, 2002

12. L Mangu, E Brill, A Stolcke, Finding consensus in speech recognition: word error minimization and other application of confusion network. Comput. Speech Lang. **14**(4), 373–400 (2000)

13. WM Campbell, JP Campbell, DA Reynolds, DA Jones, TR Leek, in *Phonetic Speaker Recognition with Support Vector Machines*, ed. by S Thrun, L Saul, B Scholkopf, in *Advances in Neural Information Processing System*, Vol. 16 (MIT Press, Cambridge, MA, 2004), pp. 1377–1384

14. D Povey, Discriminative training for large vocabulary speech recognition. PhD thesis, Cambridge University Engineering Department, 2004

15. R Collobert, S Bengio, Support vector machines for large-scale regression problems. J. Mach. Learn. Res. **1**, 143–160 (2001)

16. Y Deng, Research on the PPRVSM system for language recognition. PhD thesis, Tsinghua University, 2011

Figure 1. Integrated LRE/LVCSR, architecture and work flow. This figure illustrates the proposed embedded LRE architecture and a typical work flow for a bilingual application.

Figure 2. Example of time-aligned phone lattice. (a) Word lattice; **(b)** word-based phone lattice from the same utterance as (a); **(c)** time alignment of (b).

Figure 3. Example of phoneme-expanded 1-best lattice. (a) Initial 1-best phone transcription; **(b)** Expanded phone lattice with N-best phone hypotheses ($N = 5$). The utterance used for this example is the same as the one in Figure 2.

Figure 4. Example of frame-expanded N-best lattice (N = 5). The utterance used for this example is the same as the one in Figures 2 and 3. Since the entire lattice is huge, only the part of the first word "DONG1" is drawn.

Figure 5. Diagram of a standard PRVSM system.

Figure 6. Framework of PPRVSM system with three frontend phone recognizers.

Figure 7. DET plots of different lattice reconstruction algorithms with Mandarin frontend. Man. - Mandarin frontend.

Figure 8. EERs of A3 system as a function of percentage of links remaining after the pruning.

Figure 9. DET plots of parallel frontend A0 and A3 systems. Man. - Mandarin frontend, Eng. - English frontend, P. - parallel frontend.

Figure 10. EERs of parallel frontend systems of different test utterance lengths.

Table 1. Utterance length distribution of the test data set

Utterance length	**<1 s**	**1–2 s**	**2-3 s**	**3-4 s**	**4–5 s**	**5–6 s**	**6–7 s**	**7–8 s**	**>8 s**
Number of utterances	3,549	4,114	2,651	1,674	825	425	228	120	738

Table 2. Experimental configurations

Backend ID	Frontend type	Lattice reconstruction algorithm for VSM training and test
A0	LVCSR	None (using word-based phone lattices)
A1	LVCSR	Time-aligned phone lattice
A2	LVCSR	Phoneme-expanded 1-best lattice
A3	LVCSR	Frame-expanded N-best lattice
A3'	LVCSR	Pruned frame-expanded N-best lattice
PRVSM	Phone-loop	None (standard PRVSM system)
PRVSM-alt	Phone-loop	None (standard PRVSM system with LVCSR acoustic model)

Each experiment was implemented twice, once with an English LVCSR frontend and once with a Mandarin LVCSR frontend

Table 3. Comparison of EER for Mandarin and English frontends and different lattice reconstruction algorithms

Lattice reconstruction algorithm	**Mandarin EER (%)**	**English EER (%)**
A0. Word-based phone lattice	25.00	28.80
A1. Time-aligned phone lattice	21.69	22.75
A2. Phoneme-expanded 1-best lattice	21.43	21.71
A3. Frame-expanded N-best lattice	19.26	20.64
A3'. Pruned frame-expanded N-best lattice	18.94	20.35
PRVSM, standard PRVSM system	17.24	18.90
PRVSM-alt, PRVSM with LVCSR acoustic model	18.73	19.93

EERs of the A3' system were obtained by using the 84% pruning beam width for Mandarin and 81% for English

Table 4. Performance of parallel frontends

Lattice reconstruction algorithm	EER (%)
A0. Word-based phone lattice	21.28
A1. Time-aligned phone lattice	15.89
A2. Phoneme-expanded 1-best lattice	15.79
A3. Frame-expanded N-best lattice	15.40
A3'. Pruned frame-expanded N-best lattice	14.98
PRVSM, standard PRVSM system	14.55
PRVSM-alt, PRVSM with LVCSR acoustic model	14.97

EERs of the A3' system were obtained by using the 84% pruning beam width for Mandarin and 81% for English

Table 5. Additional time costs of the lattice reconstruction algorithms and LRE

Lattice reconstruction algorithm	Real time factor
A1. Time-aligned phone lattice	0.19
A2. Phoneme-expanded 1-best lattice	0.26
A3. Frame-expanded N-best lattice	0.07
A3'. Pruned frame-expanded N-best lattice	0.07
PRVSM, standard PRVSM system	0.20
PRVSM-alt, PRVSM with LVCSR acoustic model	0.44

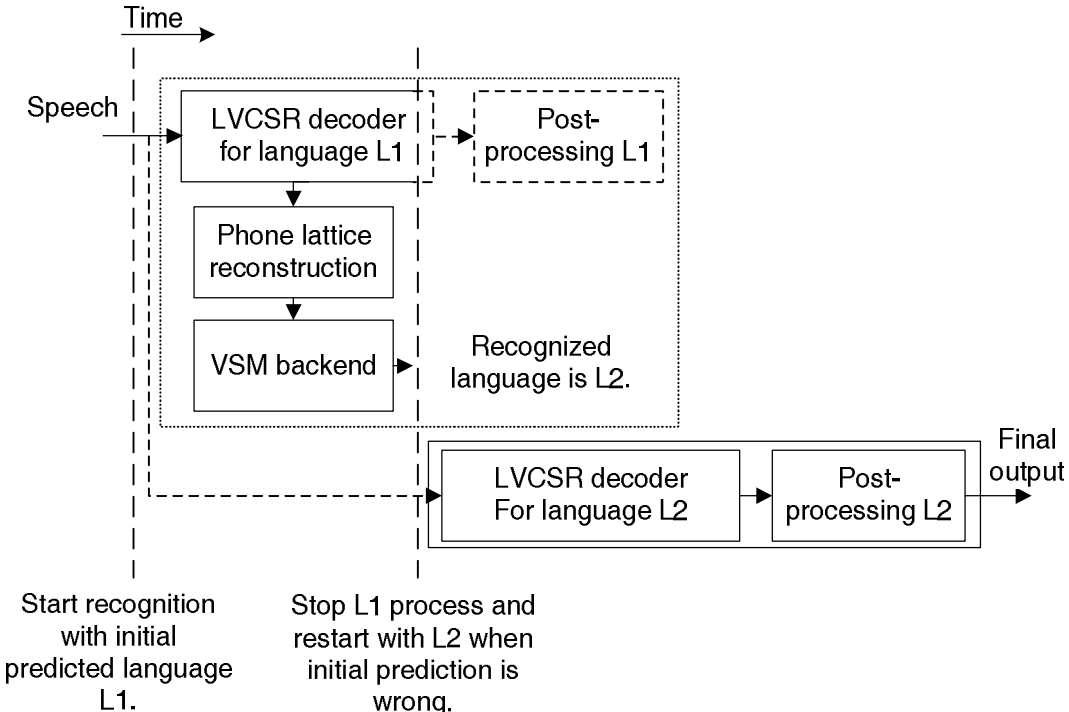
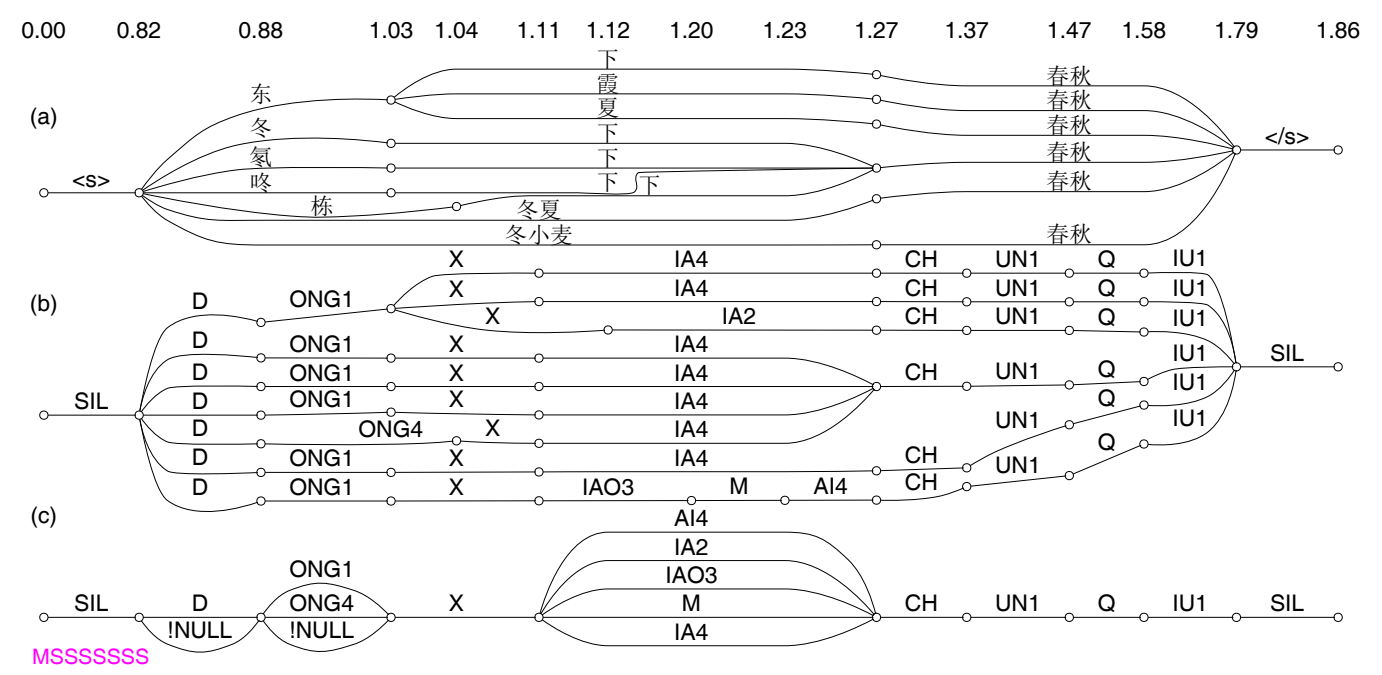
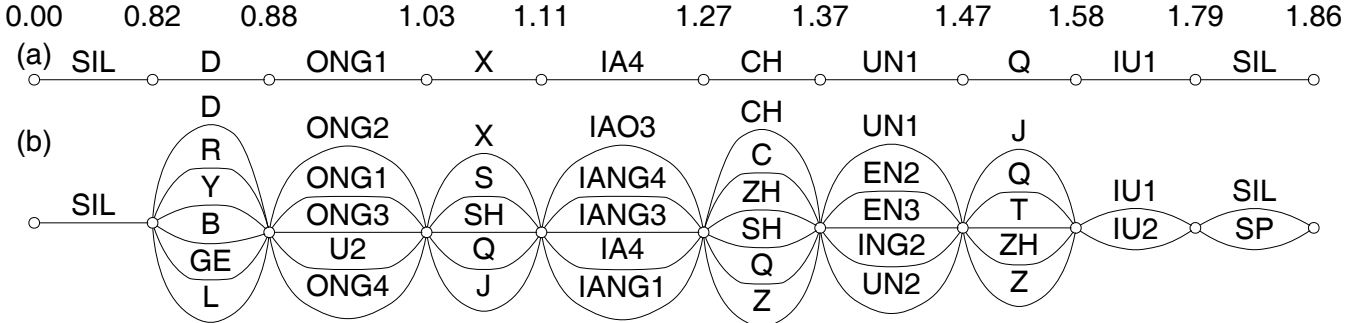
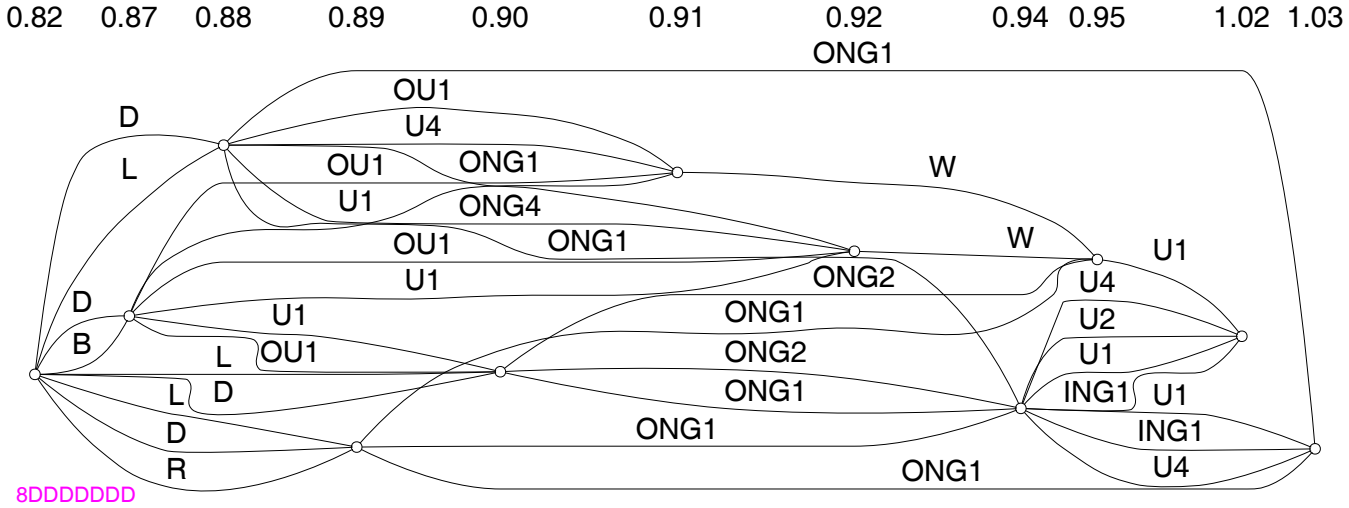

Time
Speech
LVCSR decoder for language L1
Post-processing L1
Phone lattice reconstruction
VSM backend
Recognized language is L2.
LVCSR decoder For language L2
Post-processing L2
Final output
Start recognition with initial predicted language L1.
Stop L1 process and restart with L2 when initial prediction is wrong.

Figure 1


0.00
0.82
0.88
1.03
1.04
1.11
1.12
1.20
1.23
1.27
1.37
1.47
1.58
1.79
1.86
(a)
<s>
东
冬
氢
咚
栋
下
霞
夏
下
下
下
下
冬夏
冬小麦
春秋
春秋
春秋
春秋
春秋
春秋
</s>
(b)
SIL
D
ONG1
ONG4
X
IA4
IA2
IAO3
M
AI4
CH
UN1
Q
IU1
SIL
(c)
SIL
D
!NULL
ONG1
ONG4
!NULL
X
AI4
IA2
IAO3
M
IA4
CH
UN1
Q
IU1
SIL
MSSSSSSS


0.00
0.82
0.88
1.03
1.11
1.27
1.37
1.47
1.58
1.79
1.86
(a)
SIL
D
ONG1
X
IA4
CH
UN1
Q
IU1
SIL
(b)
SIL
D
R
Y
B
GE
L
ONG2
ONG1
ONG3
U2
ONG4
X
S
SH
Q
J
IAO3
IANG4
IANG3
IA4
IANG1
CH
C
ZH
SH
Q
Z
UN1
EN2
EN3
ING2
UN2
J
Q
T
ZH
Z
IU1
IU2
SIL
SP
4SSSSSSS


0.82
0.87
0.88
0.89
0.90
0.91
0.92
0.94
0.95
1.02
1.03
ONG1
OU1
U4
D
L
OU1
ONG1
W
U1
ONG4
OU1
ONG1
W
U1
U1
ONG2
U4
D
U1
ONG1
U2
B
L
OU1
ONG2
U1
L
D
ONG1
ING1
U1
D
ONG1
ING1
R
ONG1
U4
8DDDDDDD

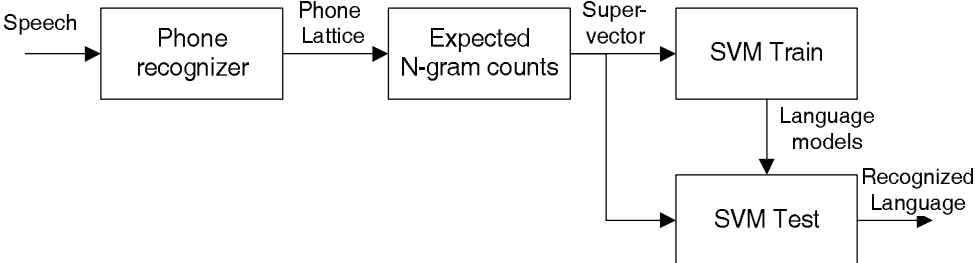

Speech
Phone recognizer
Phone Lattice
Expected N-gram counts
Super-vector
SVM Train
Language models
SVM Test
Recognized Language

Figure 5

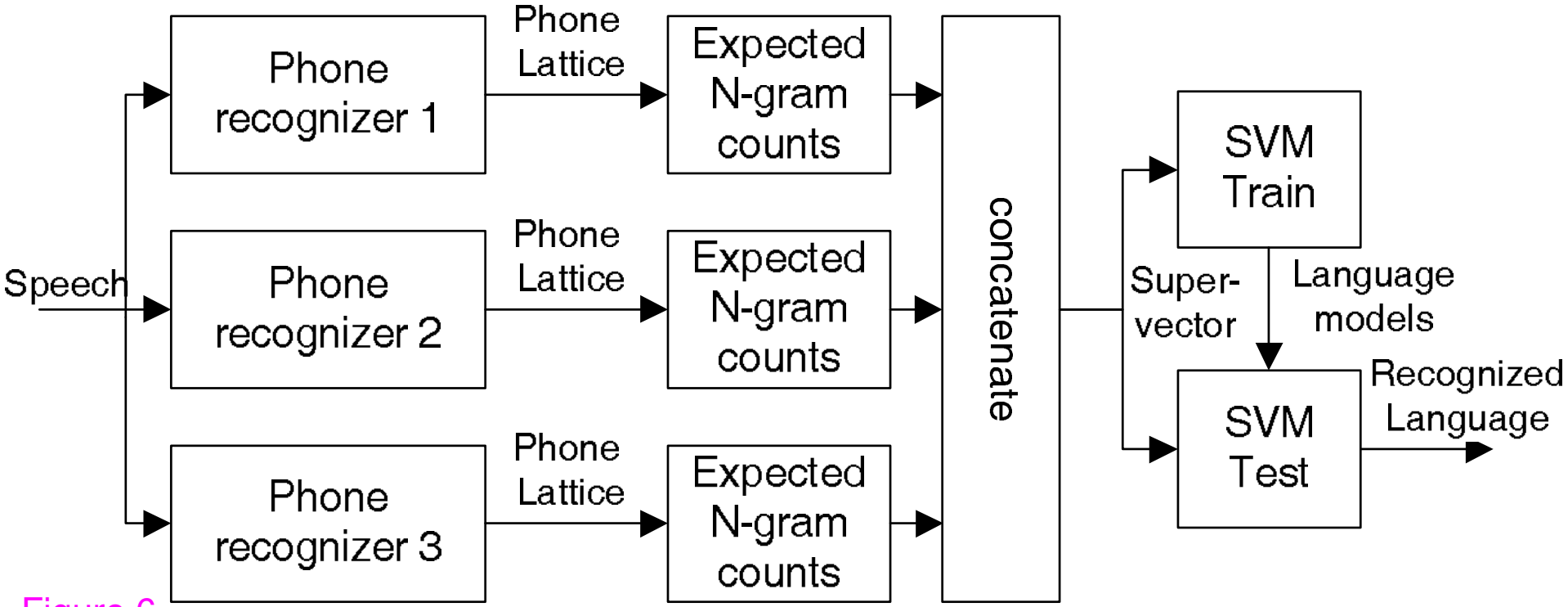

Speech
Phone recognizer 1
Phone recognizer 2
Phone recognizer 3
Phone Lattice
Phone Lattice
Phone Lattice
Expected N-gram counts
Expected N-gram counts
Expected N-gram counts
concatenate
Super-vector
SVM Train
Language models
SVM Test
Recognized Language

Figure 6

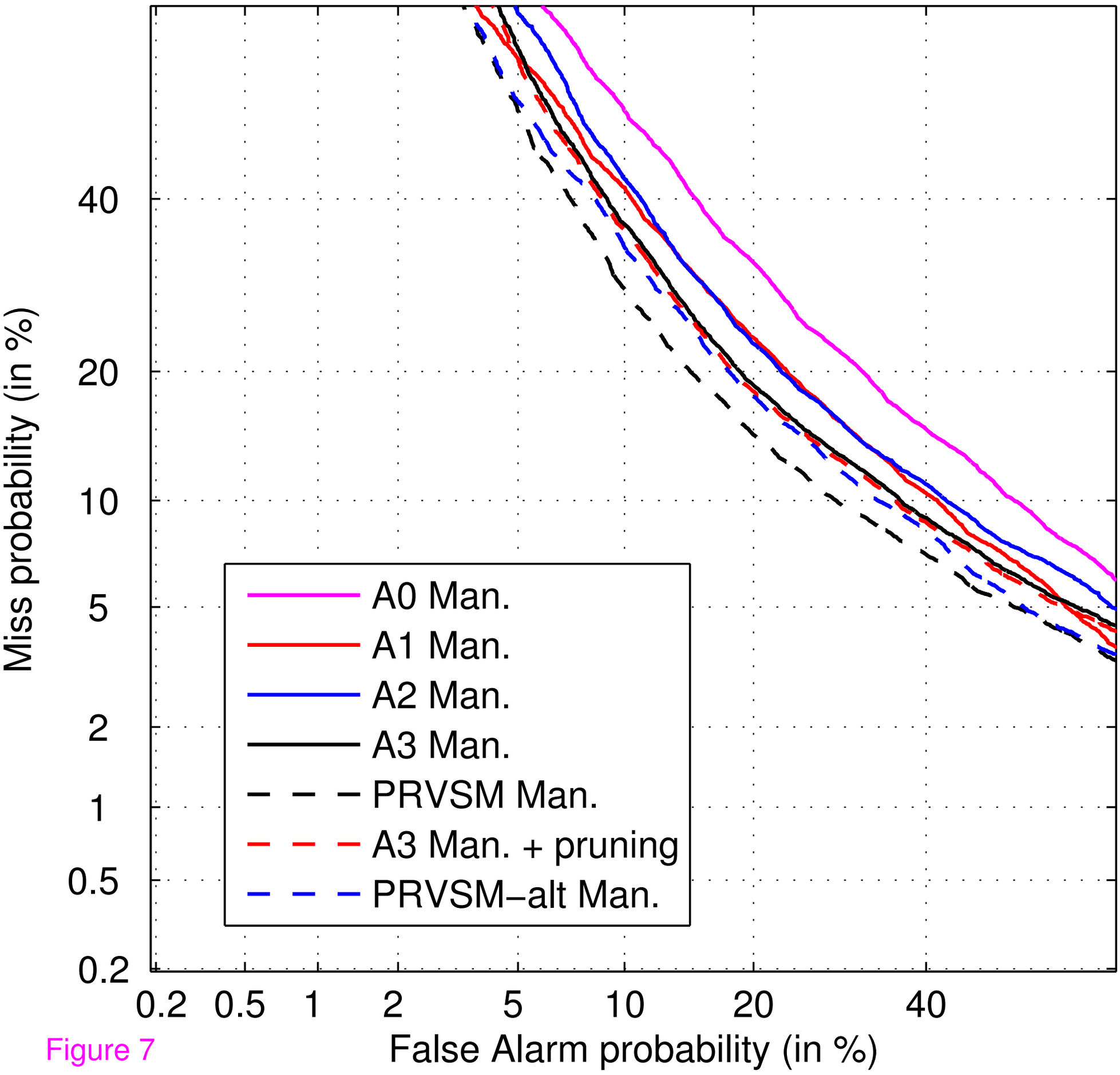

Miss probability (in %)
40
20
10
5
2
1
0.5
0.2
0.2
0.5
1
2
5
10
20
40
False Alarm probability (in %)
A0 Man.
A1 Man.
A2 Man.
A3 Man.
PRVSM Man.
A3 Man. + pruning
PRVSM–alt Man.

Figure 7

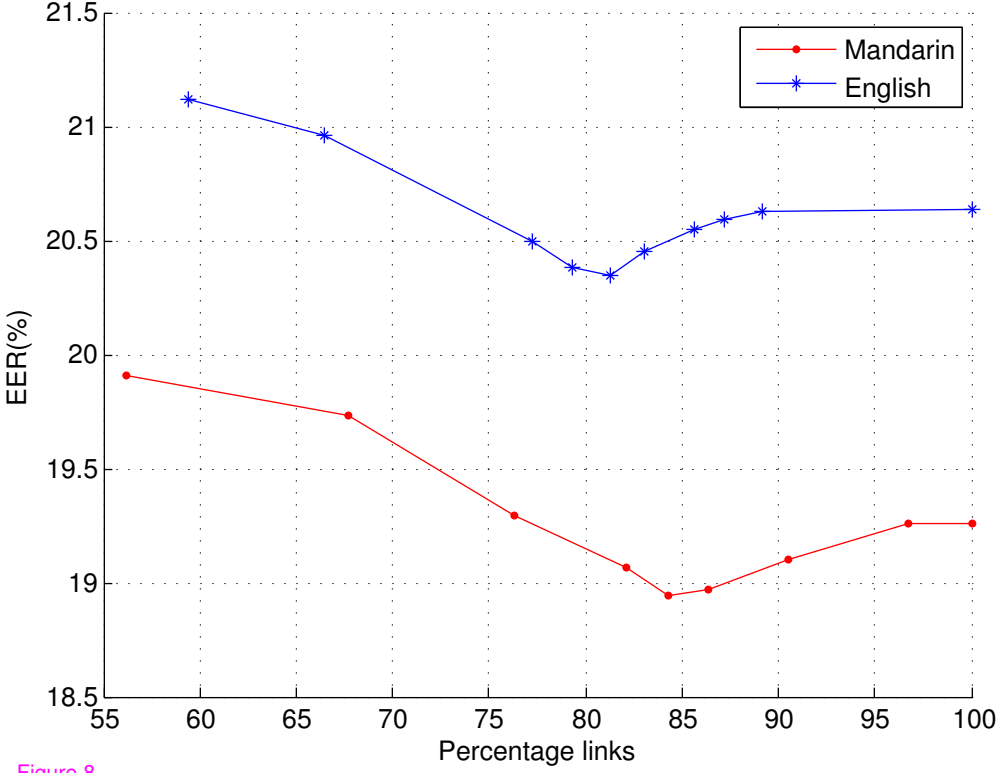

Mandarin
English
EER(%)
Percentage links
21.5
21
20.5
20
19.5
19
18.5
55
60
65
70
75
80
85
90
95
100

Figure 8

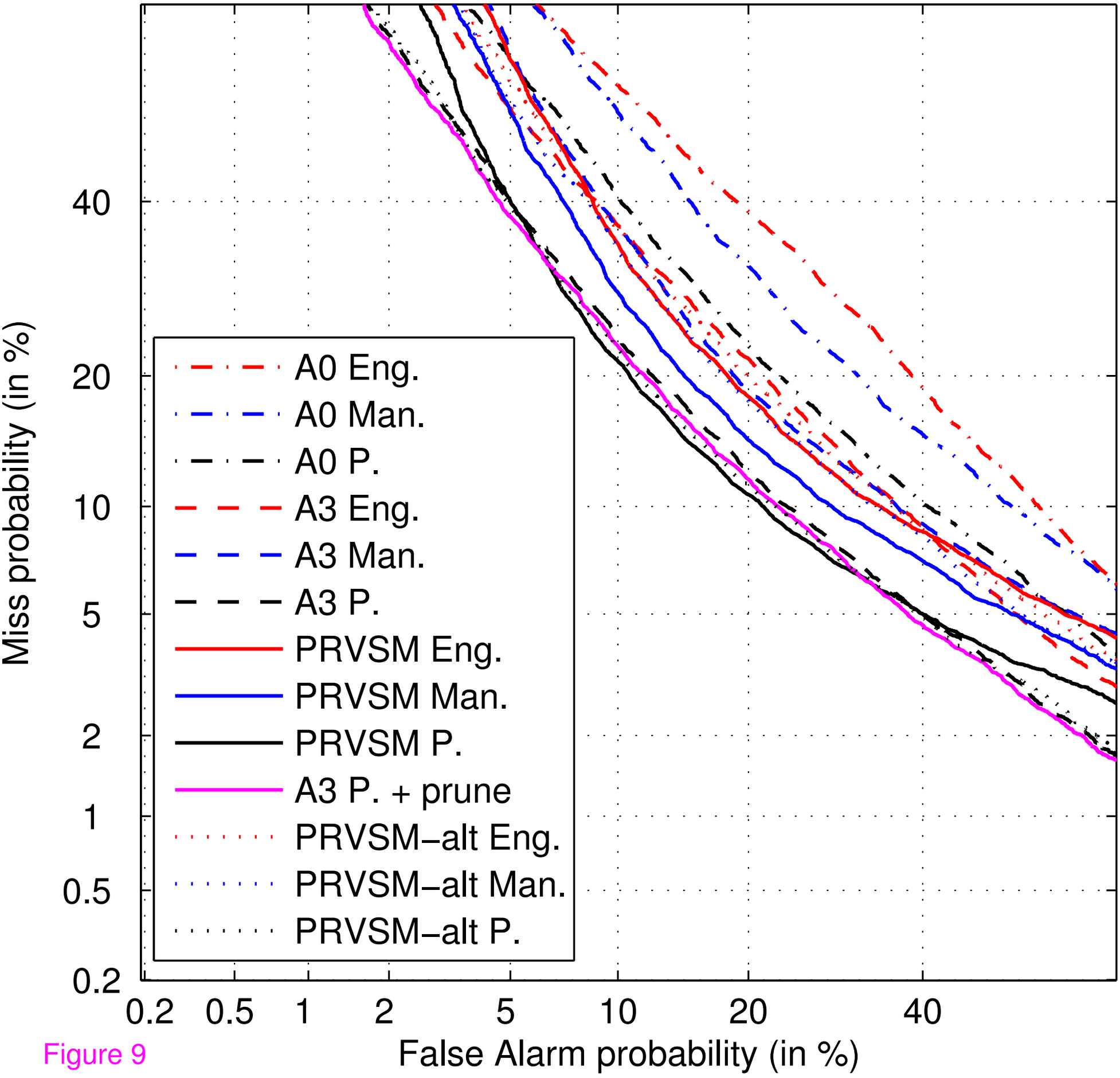

Miss probability (in %)
40
20
10
5
2
1
0.5
0.2
0.2
0.5
1
2
5
10
20
40
False Alarm probability (in %)
A0 Eng.
A0 Man.
A0 P.
A3 Eng.
A3 Man.
A3 P.
PRVSM Eng.
PRVSM Man.
PRVSM P.
A3 P. + prune
PRVSM–alt Eng.
PRVSM–alt Man.
PRVSM–alt P.

Figure 9

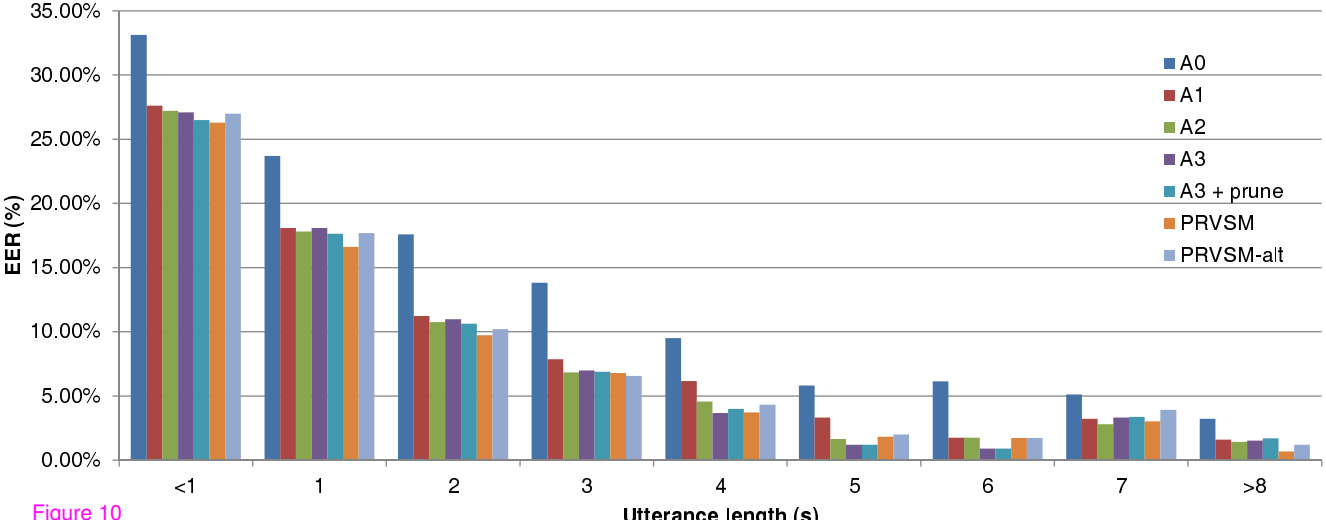

35.00%
30.00%
25.00%
20.00%
15.00%
10.00%
5.00%
0.00%
EER (%)
A0
A1
A2
A3
A3 + prune
PRVSM
PRVSM-alt
<1
1
2
3
4
5
6
7
>8
Utterance length (s)

Figure 10