A STUDY OF LATTICE-BASED SPOKEN TERM DETECTION FOR CHINESE SPONTANEOUS SPEECH

Sha Meng[1,2]*, Peng Yu*[2]*, Frank Seide*[2]*, and Jia Liu*[1]

[1]Tsinghua National Laboratory for Information Science and Technology,
Department of Electronic Engineering, Tsinghua University, 100084 Beijing, P.R.C.
mengs04@mails.tsinghua.edu.cn, liuj@tsinghua.edu.cn
[2]Microsoft Research Asia, 5F Beijing Sigma Center, 49 Zhichun Rd., 100080 Beijing, P.R.C.
{rogeryu,fseide}@microsoft.com

ABSTRACT

We examine the task of spoken term detection in Chinese spontaneous speech with a lattice-based approach. We compare lattices generated with different units: word, character, tonal syllable and toneless syllable, and also look into methods of converting lattices from one unit to another one. We find the best system is with toneless-syllable lattices converted from word lattices. Further improvement is achieved by lattice post-processing and system combination. Our best system has an accuracy of 80.2% on a keyword spotting task.

Index Terms— speech indexing, lattice, posterior, keyword spotting

1. INTRODUCTION

Improving accessibility for the overwhelming amounts of speech data available today necessitates the development of robust Spoken Term Detection (STD, also known as keyword spotting) and Spoken Document Retrieval (SDR) techniques. To date, a significant amount of work in this field has been done for English.

The TREC (Text Retrieval Conference) SDR track has fostered research on audio retrieval of broadcast-news clips. Most benchmarking systems applied text retrieval directly on the speech recognition transcripts. Due to the low Word Error Rate (WER) for broadcast news (<20%) and high redundancy of queries in documents, a similar accuracy compared to a human reference was achieved and thus was considered a "solved problem"[1]. However, further research [2, 3, 4] found that this approach does not apply to spontaneous speech, where typical WER is about 40% to 50%. In this situation, indexing recognition alternates, normally represented as lattices, provides significant improvement.

Among research on lattice-based indexing for English, there have been discussions about the choice of indexing unit between word and sub-word (normally phoneme for English). Word-based systems, usually based on Large-Vocabulary Continuous Speech Recognizers (LVCSR), suffer from the Out-Of-Vocabulary (OOV) problem, which is more serious for STD tasks as queries chosen by users tend to be rare words, and have a higher probability to be OOV words. Phoneme-based systems have no OOV problem, but have significantly lower precision due to weaker language models. [5] found that a phonetic system outperformed a word-based one on a recall-emphasized task. [6] showed significant improvement by combining two systems.

The same requirement for STD and SDR systems rises for Chinese. The (Mandarin) Chinese language has several distinctive characteristics compared to English. Chinese words are graphemically made up of characters. The set of most-common 6000 Chinese characters has a sufficient coverage for most user scenarios. Thus the OOV problem of LVCSR system is significantly reduced as those characters themselves are in the dictionary as well. Additionally, the Chinese language has a closed set of Constant-Vowel structured syllables. As syllables usually have a more stable segmentation and a closer link to semantic-level information, it provides a better choice of sub-word unit than phonemes.

Previous research [7, 8, 9] on Chinese spoken term detection used lattice-based approach as previously done for English . Both of them built systems with syllables.

In this paper, we also examine lattice-based spoken term detection for Chinese spontaneous speech. Specially, we look into the problem of comparing lattices with word and different sub-word units (character, tonal and toneless syllable). In addition, we also discuss methods to convert lattices generated with higher-level unit, i.e., more semantic based like word, to lower-level unit, i.e. more phonetic based like syllable. It is observed that the best performance comes from toneless-syllable lattices converted from word lattices. Further improvement is achieved by lattice post-processing method and system combination. Our best system has an FOM of 80.2%,

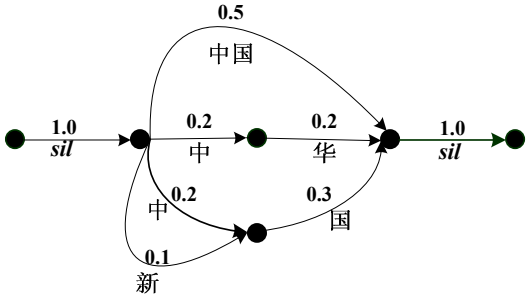
Work performed during the 1st author's internship at MSR Asia.

compared with 51.5% from using speech recognition transcripts only.

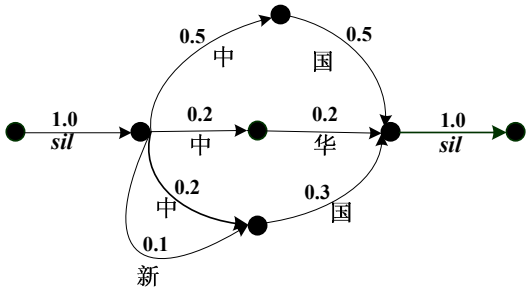
The rest of paper is organized as follows. Section 2 introduces the lattice-based spoken term detection algorithm. Section 3 describes methods of lattice generation and conversion with different unit. Section 4 and section 5 discuss methods for lattice post-processing and system combination. Section 6 shows the results and section 7 concludes the whole paper.

2. LATTICE-BASED WORD SPOTTING

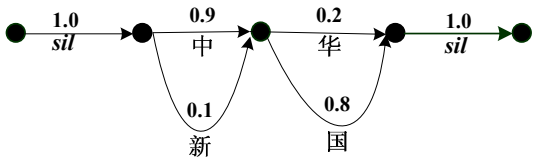
In this section, we give a brief introduction to lattice-based spoken term detection.



(a) Original posterior lattice



(b) Converting word lattice to character lattice



(c) Lattice after TMI-node merging

Fig. 1. Examples of Posterior Lattice: (a) is a posterior lattice with word arcs. Splitting the word arc into character ones, result lattice is shown in (b). Merging the lattice with TMI-node method, (c) shows the final result.

2.1. Lattice Definition

A lattice $\mathcal{L} = (\mathcal{N}, \mathcal{A}, n_{\text{start}}, n_{\text{end}})$ is a directed acyclic graph (DAG) with $\mathcal{N}$ being the set of nodes, $\mathcal{A}$ is the set of arcs, and n_{start}, $n_{\text{end}} \in \mathcal{N}$ being the unique initial and unique final node, respectively.

Each node $n \in \mathcal{N}$ has an associated time $t[n]$ and possibly an acoustic or language-model context condition. Arcs are 4-tuples $a = (S[a], E[a], I[a], w[a])$. $S[a]$, $E[a] \in \mathcal{N}$ denote the start and end node of the arc. $I[a]$ is the word (or subword) identity. Last, $w[a]$ shall be a weight assigned to the arc by the recognizer. Specifically, $w[a] = p_{\text{ac}}(a)^{1/\lambda} \cdot P_{\text{LM}}(a)$ with acoustic likelihood $p_{\text{ac}}(a)$, LM probability P_{LM}, and LM weight λ. [1] Normally the recognizer will also provide the best pronunciation for each arc, when multiple pronunciations exists for word $I[a]$.

In addition, we define *paths* $\pi = (a_1, \cdots, a_K)$ as *sequences* of connected arcs. We use the symbols S, E, I, and w for paths as well to represent the respective properties for entire paths, i.e. the path start node $S[\pi] = S[a_1]$, path end node $E[\pi] = E[a_K]$, path label sequence $I[\pi] = (I[a_1], \cdots, I[a_K])$, and total path weight $w[\pi] = \prod_{k=1}^{K} w[a_k]$.

2.2. Posterior Lattice Representation

It was found in [4] that an alternative but equivalent representation of lattice, which we call *posterior lattice*, is more convenient in many cases.

For the posterior lattice, we define the *arc posteriors* $P_{\text{arc}}[a]$ and *node posteriors* $P_{\text{node}}[n]$ as

$$P_{\text{arc}}[a] \;=\; \frac{\alpha_{S[a]} \cdot w[a] \cdot \beta_{E[a]}}{\alpha_{n_{\text{end}}}} \;\;;\;\; P_{\text{node}}[n] = \frac{\alpha_n \cdot \beta_n}{\alpha_{n_{\text{end}}}},$$

with *forward-backward probabilities* α_n, β_n defined as:

$$\alpha_n \;=\; \sum_{\pi: S[\pi]=n_{\text{start}} \wedge E[\pi]=n} w[\pi] \;\;;\;\; \beta_n \;=\; \sum_{\pi: S[\pi]=n \wedge E[\pi]=n_{\text{end}}} w[\pi]$$

α_n and β_n can be conveniently computed using the well-known forward-backward recursion, e.g. [10].

The posterior lattice representation stores four fields with each edge: $S[a]$, $E[a]$, $I[a]$, and $P_{\text{arc}}[a]$, and two fields with each node: $t[n]$, and $P_{\text{node}}[a]$.

In our previous work [3] it was shown that in a word spotting task, ranking by the posterior probability of phrase is theoretically optimal. With the posterior lattice representation, the phrase posterior of query string Q is computed as

$$\begin{aligned} & P(*, t_s, Q, t_e, * | O) \\ = & \sum_{\substack{\pi=(a_1,\cdots,a_K): \\ t[S[\pi]]=t_s \wedge t[E[\pi]]=t_e \wedge I[\pi]=Q}} \frac{P_{\text{arc}}[a_1] \cdots P_{\text{arc}}[a_K]}{P_{\text{node}}[S[a_2]] \cdots P_{\text{node}}[S[a_K]]}. \end{aligned} \tag{1}$$

[1] Despite its name, the function of the LM weight is now widely considered to flatten acoustic emission probabilities. This matters when sums of path probabilities are taken instead of just determining the best path.

The posterior representation is lossless. It has several advantages as stated in [4]. In our case, the posterior lattice representation makes it easy to segment an arc with longer unit (e.g. word) to multiply short units (e.g. characters), and to merge multiple arcs and nodes together. This will be shown in later sections.

3. LATTICE GENERATION AND SEARCH WITH WORD AND SUB-WORD UNITS

Compared with English, the Chinese language has several distinctive characteristics:

- Graphemically, a *word* is a sequence of *characters*. Though number of words is unlimited as in English, the 6000 common characters have sufficient coverage for most user scenarios. A character alone is a word itself, and has its own meaning.
- Phonetically, a *character* is a Consonant-Vowel structured *syllable*. In total, there are about 420 base syllables. Chinese is a tonal language, for each base syllable, there are up to five tone types.
- Sometimes one character may map to multiple tonal syllables in different context of words. Those characters are called polyphonies. Polyphonies are common in Chinese. On average each character has about 1.2 pronunciations.

For the unit of lattice generation, there are five choices: word, character, tonal-syllable, toneless-syllable, phoneme. Phonemes are typically not used in Chinese speech recognition as they has no real benefit over syllables.

3.1. Lattice Generation and Matching

A LVCSR decoder is used to generate word-based lattices. As the characters themselves are in the vocabulary as well, there is nearly no OOV problem. At search time, a standard ngram-based word breaker is used to parse a query into word sequence. The word breaker uses the same dictionary as the recognizer, which means no OOV word will appear in the parsed word sequence. A query is considered a match when the word sequence matches consecutive arcs in a lattice.

Same decoder is used to generate lattices for characters, tonal syllables and toneless syllables, but with trigram language models for characters, tonal syllables and toneless syllables respectively. At search time, for character system, a query is directly broken into characters and matched against lattices. For syllables, the word breaker is first used to parse queries, and then the dictionary is used to lookup the tonal or toneless pronunciation of words. If multiple pronunciations exist for words, they will all be matched against lattices.

3.2. Converting Word Lattices To Character Lattices

Though word lattices do not have OOV problem, it suffers from the ambiguity of word breaking. E.g., phrase "中国人" ("Chinese People") can be broken into either "中国人" (single word) or "中国-人" (two words). Thus an arc in lattice with "中国人" will not match query "中国" ("Chinese") though it should.

The problem could be solved by converting word lattices to character lattices, and matching queries by characters only. For each arc a in lattice, $I[a] = W = (c_1, \cdots, c_N)$, where W is the word with the arc, and c_i are consecutive characters, the conversion could be easily done with posterior lattice representation in following steps:

- create new nodes $n_1, \cdots, n_{N-1}$, with

$$\begin{aligned} t[n_i] &= \frac{(i * t[E[a]] + (N-i) * t[S[a]])}{N}; \\ P_{\text{node}}[n_i] &= P_{\text{arc}}[a]; \end{aligned}$$

- create new arcs $a_1, \cdots, a_N$, with

$$S[a_i] = \begin{cases} S[a] : i = 1 \\ n_{i-1} : i > 1 \end{cases}; \qquad E[a_i] = \begin{cases} n_i & : i < N \\ E[a] : i = N \end{cases};$$
$$P_{\text{arc}}[a_i] = P_{\text{arc}}[a]; \qquad I[a_i] = c_i;$$

- delete a.

Fig. 1 depicts this process. Fig. 1(a) is a word lattice, while (b) is a converted character lattice.

3.3. Converting Word and Character Lattices To Syllable Lattices

The reason we want to convert word or character lattices to syllable lattices is to tolerate recognizer errors with homophones (words or characters having same pronunciations), i.e., words or characters in lattices will be counted as matches as long as they have same pronunciations as queries. The conversion is straightforward by replacing word or character labels on each arc to corresponding pronunciations (i.e. syllables). For word lattice, the same arc splitting algorithm as in 3.2 is required. Sometimes words and characters have multiple pronunciations (with polyphonies), in this case, the best pronunciation info for each arc provided by the recognizer is used.

4. LATTICE POST-PROCESSING

In result lattices from recognizers, nodes are not only time points, but also carry language model context information. This means that, at the same time point, there could exist multiple nodes. If we pinch these nodes together, we add additional paths in lattice, which will result in a better recall for

phrase matches. Though at the same time we are generating false positives, experimental results show that it is not harmful for keyword spotting tasks, as most of false positives are random combinations which will not used as queries by users.

To achieve this, we proposed to use the TMI (Time-based Merging for Indexing) processing first presented in [4], which is proposed for reducing lattice size. But the algorithm fits here as well for increasing lattice recall.

The TMI method starts from a simple criterion: reduce the number of nodes as much as possible by merging consecutive nodes without generating loop edges. It turns out that the optimal merging could be achieved by a Dynamic Programming algorithm as below:

- merge all nodes with same time stamps, with node posteriors summed up;
- line up all nodes in ascending time order, $t[n_i] < t[n_{i+1}]$, $i = 1, \cdots, N$;
- for each node n, find out the furthest node that it can be grouped without generating loop arcs, denoting its index as $T[n]$;
- set group count $C[n_1] = 1$; $C[n_i] = \infty$, $i > 0$. Here $C[n_i]$ means the minimal number of nodes after node merging with sub-lattice containing $n_1, \cdots, n_i$ and connecting arcs;
- set backpointer $B[n_1] = -1$; $B[n_i] = n_i$, $i > 0$;
- for $i = 1, \cdots, N-1$:
  - for $j = i+1, \cdots, T[n_i]$: if $C[n_{j+1}] > C[n_i]+1$:
    - $C[n_{j+1}] = C[n_i] + 1$;
    - $B[n_{j+1}] = n_i$; // clustering node $n_i, \cdots, n_j$
- trace back and merge nodes:
  - set $k = N$, repeat until $k = -1$:
    - merge nodes from $B[n_k]$ to n_{k-1} with node posteriors summed up;
    - $k = B[n_k]$.
- merge all edges with same start node and end node, with edge posteriors summed up.

Fig. 1(c) shows an example of TMI processing.

5. SYSTEM COMBINATION

[6, 2, 11] show significant improvement from combining word-level lattices with phonetic lattices for English keyword spotting tasks. If there are N systems, with query Q and time slot (t_s, t_e), the i-th system has a phrase posterior of $P^i(*, t_s, Q, t_e, *|O)$, a combined system will have:

$$P^{\text{comb}}(*, t_s, Q, t_e, *|O) = \sum_i \gamma_i \cdot P^i(*, t_s, Q, t_e, *|O),$$

where $\sum_i \gamma_i = 1$. Ideally, the weights γ_i should be tuned on a development set to get best performance. However, experiments show that those weights are really not sensitive for final results. Experiments in the result section use equal weights.

6. RESULTS

6.1. Setup

We evaluate our method by a keyword-spotting task on a 4-hour long Chinese spontaneous speech set. The phone set contains 187 phones, with 28 "initial"s (the Consonant) phones, 157 tonal "final"s (the Vowel), and two silence phones. There are totally 1666 tonal syllable and 423 toneless syllables [12]. An acoustic model trained on 154-hour reading-style speech plus 148-hours spontaneous mandarin speech is used for all setups. 39-dimension MFCC is used. A dictionary with 68933 words is used for both the LVCSR recognizer and for the word breaker. Word and character trigrams are trained from a text corpus containing about 2.1 billion characters. Syllable-based trigrams are trained from all the pronunciations in the dictionary.[2]

The baseline WER of all recognizers are shown in Table 1. To compare among different units, Character Error Rate (CER), Syllable Error Rate (SER) and toneless SER are listed as well. As expected, the more semantic information a unit carries, the stronger constraint a trigram language model provides, and the lower Error Rate is obtained. Therefore the word-based recognizer has the best error rate.

Table 1. Accuracy of different recognizers (WER:word error rate, CER: character error rate, SER: syllable error rate, all in %)

decoder unit	WER	CER	SER	Toneless SER
Word	48.43	36.98	35.38	30.81
Character	–	42.90	41.33	35.90
Syllable	–	–	61.30	50.39
Toneless Syllable	–	–	–	51.93

An automatic procedure described in [5] was used to select queries. Example queries are "春节" ("spring festival"), "俄罗斯"("Russia"), "埃菲尔铁塔"("Eiffel Tower"). A statistics of number of syllables in each queries are shown in Table 2.

Table 2. Query statistics against number of syllables. #occurrences are total occurrences in reference transcripts.

#syl. in query	2	3	4	5+	Total
#queries	2004	777	846	352	3979
#occurrences	3095	930	942	379	5346

[2]Our experiments on English setup show that training phonetic trigrams on a dictionary (other than from pronunciation transcripts of a text corpus) provides satisfying performance already.

Table 3. Lattice Generation with different units. The first line is bestpath with word-based system, others are for lattices. Performance on query set with different number of syllables are listed. All numbers are in %.

#syl. in query		all		2		3		4		5+	
id	decoder unit	FOM	REC	FOM	REC	FOM	REC	FOM	REC	FOM	REC
S1b	word bestpath	<51.5	51.5	<48.1	48.1	<48.3	48.3	<61.9	61.9	<61.0	61.0
S1	word	68.3	69.2	63.8	65.4	70.2	70.4	78.0	78.0	75.5	75.5
S2	character	67.5	70.5	66.0	70.9	66.5	67.0	73.4	73.5	68.3	68.3
S3	tonal syllable	66.9	73.5	67.0	78.0	69.6	70.4	66.8	66.9	59.9	59.9
S4	toneless syllable	67.6	75.1	67.6	80.2	72.0	72.9	68.1	68.2	56.2	56.2

Results are reported in Figure of Merit (FOM), which is defined by National Institute of Science and Technology (NIST) as the detection/false-alarm curve averaged over the range of [0...10] false alarms per hour per keyword. Lattice recall (recall of all query matches within lattice, which is an upper bound of FOM) is listed as well for analysis purpose.

6.2. Lattice Generation with Different Units

Table 3 lists results for lattices generated with different recognizers. Results against number of syllables in query are listed as well.

The first and second lines compares the bestpath-only approach versus a lattice-based approach for a word-based system. At this WER level, the LVCSR bestpath has a recall at only 51.5%, while the lattice increase the recall to 69.2%, with the FOM improved to 68.3%.

The next three lines list performance of lattices generated with characters, tonal syllables and toneless syllables. They all have a better recall than the word-based system[3], but the latter has the best FOM.

Distribution of FOM over number of syllables in query shows a clear drop for two syllable-based systems, reflecting the limited prediction scope of syllable trigrams. In contast, distribution for word-based and character-based systems are more flat, benefit from a stronger prediction power with semantic level information.

6.3. Lattice Conversion among Different Units

Table 4 show results with lattice conversion among different units, in most cases, the conversion process results in a better FOM and recall. The improvement mainly comes from two steps:

- Conversion from word lattices to character lattices (from S1 (FOM 68.3%) to S1.1 (FOM 70.9%)). This improvement mainly comes from removing the ambiguity of the word breaker as discussed in section 3.2;
- removing tone information (from S1.2 to S1.3, S2.1 to S2.2, and S3 to S3.1, more than 2% FOM improvement for all). The improvement comes from tolerating the tone recognition errors from the decoder, reflecting the fact that tone information in spontaneous speech is not always stable.

[3]The lattice recall really depends on the beam setting when decoding. However, as different systems has different perplexity, it is difficult to fairly compare beam setting across systems. We tried to use the biggest beaming setting the decoder allows for each system, so the recall here could be understood as "the upper limit of recall with a realistic setup" for each system.

At the same time, conversion word lattice to tonal-syllable lattices (S1 to S1.2) is only marginally better than conversion to character lattices. Conversion from character lattices to tonal-syllable (S2 to S2.1) lattices got even worse. This is mainly caused by the ambiguity when mapping words/characters with multiple pronunciations to tonal syllables. Though the recognizer provides the best pronunciation in this case, it seems that the information is not accurate enough. This is more serious when converting character lattices to tonal-syllable lattice, as character sequence has more pronunciation variances than words.

The best setup in the table is by converting from word lattices to toneless-syllable lattices, which has an FOM at 73.3%.

Table 4. Keyword spotting results for different setups. The first four blocks show results for converting lattices to different units. Rightmost two column are after TMI post-processing. The bottom four lines show results for system combination. All numbers are in %.

		Raw Lattice		+ TMI proc.	
no.	index	FOM	Rec	FOM	Rec
S1	word	68.3	69.2	67.5	68.4
S1.1	=>character	70.9	73.2	71.7	74.0
S1.2	=>syllable	71.3	73.8	72.3	74.9
S1.3	=>toneless syl.	73.3	77.4	74.2	78.5
S2	character	67.5	70.5	68.3	71.7
S2.1	=>syllable	61.3	64.6	65.1	69.2
S2.2	=>toneless syl.	69.5	75.0	71.4	77.8
S3	syllable	66.9	73.5	68.8	76.6
S3.1	=>toneless syl.	70.8	79.4	71.8	82.0
S4	toneless syl.	67.6	75.1	69.7	78.8
C1	S1+S1.1+S1.2+S1.3			74.3	78.0
C2	S2+S2.1+S2.2			73.6	79.4
C3	S3+S3.1			71.9	82.1
C5	C1+C2+C3+S4			80.2	88.1

6.4. Lattice Post-Processing

The right-most two columns of Table 4 show the effects of TMI lattice post-processing, where we see a consistent improvement for almost all the setups for both FOM and recall except S1. This improvement is caused by a higher recall, which comes from the additional path created by node merging. The corresponding increase in FOM shows the false positives introduced by this merging is not harmful.

6.5. System Combination

The last four lines of Table 4 show results with system combination. The line C1, C2, and C3 combines the setups coming from the same recognizer, almost no improvement is observed over the best individual system. However, when combining setups from different decoders, as shown in line C5, FOM is significantly improved to 80.2%, from the best individual system S1.3 with 74.2%.

7. CONCLUSIONS

This paper has examined lattice-based spotaneous Chinese speech. In order to better capture the characteristics of the Chinese language, we used different units for lattice generation, including word, character, tonal and toneless syllables. We also looked into converting word lattices to character or syllable lattices, and character lattices to syllables lattices. Overall it was found that the best performance comes from toneless-syllable lattices converted from word lattices, which achieved an FOM of 73.3% on spontaneous Chinese speech.

Lattice post-processing were then applied aimed at creating additional links in lattice, which resulted in a higher recall. Results show that the post-processing gave about 1% improvement to FOM. Different setups were then combined together by interpolating query phrase posteriors from each setup, and the best combination achieved an FOM of 80.2%.

8. REFERENCES

[1] J. Garofolo, TREC-9 Spoken Document Retrieval Track. National Institute of Standards and Technology, `http://trec.nist.gov/pubs/trec9/sdrt9_slides/sld001.htm`.

[2] M. Saraclar, R. Sproat, Lattice-based Search for Spoken Utterance, *Proc. HLT'2004*, Boston, 2004

[3] P. Yu, K. J. Chen, C. Y. Ma, F. Seide, Vocabulary-independent Indexing of Spontaneous Speech, IEEE transaction on Speech and Audio Processing, Vol.13, No.5, Special Issue on Data Mining of Speech, Audio and Dialog.

[4] Z. Y. Zhou, P. Yu, C. Chelba, F. Seide, Towards Spoken-document Retrieval for the Internet: Lattice Indexing For Large-Scale Web-search Architectures, *Proc. HLT'2006*, New York, 2006.

[5] F. Seide, P. Yu, *et al.*, Vocabulary-independent Search in Spontaneous Speech, *Proc. ICASSP'2004*, Montreal, 2004.

[6] P. Yu, Frank Seide, A Hybrid Word/Phoneme-based Approach for Improved Vocabulary-independent Search in Spontaneous Speech, *Proc. ICLSP'2004*, Korean, 2004.

[7] B. R. Bai, B. Chen, H. M. Wang, A Multi-phase Approach for Fast Spotting of Large Vocabulary Chinese Keywords from Mandarin Speech Using Prosodic Information, *Proc. ICASSP'1997*, Munich, 1997.

[8] H. M. Wang, Experiments in Syllable-based Retrieval of TV News Speech in Mandarin Chinese, Speech Communication, 32(1-2), pp. 49-60, Sept. 2000

[9] B. Chen, H. M. Wang, L. S. Lee, Retrieval of Broadcast News Speech in Mandarin Chinese Collected in Taiwan Using Syllable-level Statistical Characters, *Proc. ICASSP'2000*, Istanbul, 2000.

[10] F. Wessel, R. Schluter, K. Macherey, and H. Ney, Confidence Measures for Large Vocabulary Continuous Speech Recognition, IEEE transaction on Speech and Audio Processing, Vol.9, No.3, Mar.2001.

[11] S. W. Lee, K. Tanaka, Y. Itoh, Combining Multiple Subword Representations for Open-vocabulary Spoken Document retrieval,*Proc. ICASSP'2005*, Philadelphia, 2005.

[12] C. Huang, Y. Shi, J. L. Zhou, M. Chu, T. Wang, E. Chang, Segmental Tonal Modeling for Phone Set Design in Mandarin LVCSR, *Proc. ICASSP'2004*, Montreal, 2004.