Spoken Term Detection Evaluation Plan

http://www.nist.gov/speech/tests/std/2006/docs/STD-announce-v6.pdf
http://nist.gov/speech/tests/std/2006/docs/std06-evalplan-v10.pdf

Patty Liu

Introduction

- Information retrieval is a major economic activity around the world, and digitized speech from many sources is growing rapidly in volume.
- Responding to the opportunity and need, NIST is designing a new evaluation initiative called Spoken Term Detection (STD).

The Task (1/3)

- The STD task is to find all of the occurrences of a specified term in a given corpus of speech data. For the STD task, a term is a sequence of one or more words. No terms will include more than five words.
- Systems must be implemented in two phases: indexing and searching. In the indexing phase, the system must process the speech data without knowledge of the terms. In the searching phase, the system uses the terms, the index, and optionally the audio to detect term occurrences.

The Task (2/3)

I. Term

- Ideally, a term would have a single specific interpretation or meaning. In order to make the implementation of STD evaluation feasible, however, the occurrence of a term in the corpus will be judged solely on the orthographic transcription of the corpus.

  Ex: “wind” (moving air) v.s. “wind” (twist) → match

  “grasshoppers” v.s. “grasshopper” → not match

  “cat” v.s. “catalog” → not match

The Task (3/3)

II. System output

- For each term supplied to the system, all of the occurrences of that term in the test corpus are to be found and statistics for each found occurrence are to be output. For each found occurrence of the given term, the system is to output a record that includes：
  - the location of the term in the audio recording
  - a score indicating how likely the term exists with more positive values indicating more likely occurrences
  - a hard (binary) decision as to whether the detection is correct

The Data (1/3)

- These are namely a development set (“DevSet”) that participating sites may use to aid their research, and an evaluation set (“EvalSet”) that will be supplied at the beginning of the formal evaluation.
- In addition, a third “pilot” evaluation data set will be created as a subset of the DevSet and provided to participants for the pilot (“Dry Run”) evaluation.

I . The Search Terms：

There will be a wide variety of search terms: single-word and multi-word terms, common and rare terms.

There will be both a total of about 1000 terms, per language, in the DevSet and the EvalSet.

The Data (2/3)

II. The Speech Corpora ：

- Three languages : Arabic (Modern Standard and Levantine), Chinese (Mandarin) , and English (American)
- The three source types : Conversational Telephone Speech (CTS), Broadcast News (BNews), and Conference Room (CONFMTG) meetings i.e., goal oriented, small group, roundtable meetings.

Table 1 Language/Source Type pairs to be tested and the durations of indexed audio for both the DevSet and EvalSet

	Arabic	Chinese	English
Broadcast News	MSA ~1 hour	Mandarin ~1 hour	American ~ 3 hours
Telephone Conversations	Levantine ~1 hour	Mandarin ~1 hour	American ~3 hours
Roundtable Meetings	No	No	American ~2 hours

The Data (3/3)

III. Forbidden Data

- The STD evaluation corpora will include speech from the following previously used RT evaluation corpora:

  - The Fall 2004 BNews and CTS Rich Transcription Evaluation corpus

  - The Spring 2006 Meeting Domain Rich Transcription evaluation corpus

- Therefore, participants who possess these corpora must refrain from examining or using them in any way for lexicon building, system training, or development testing.
- Additionally, news-oriented material (audio, textual, etc.) generated after the beginning of the current test epoch (beginning December 1, 2003) or material (other than the RT03 eval data) from the DevSet epoch (February 2001) may not be used in any way for system development or training.

Implementation Details (1/4)

Figure 1 shows the system input/output files and how they relate to system operation and evaluation.

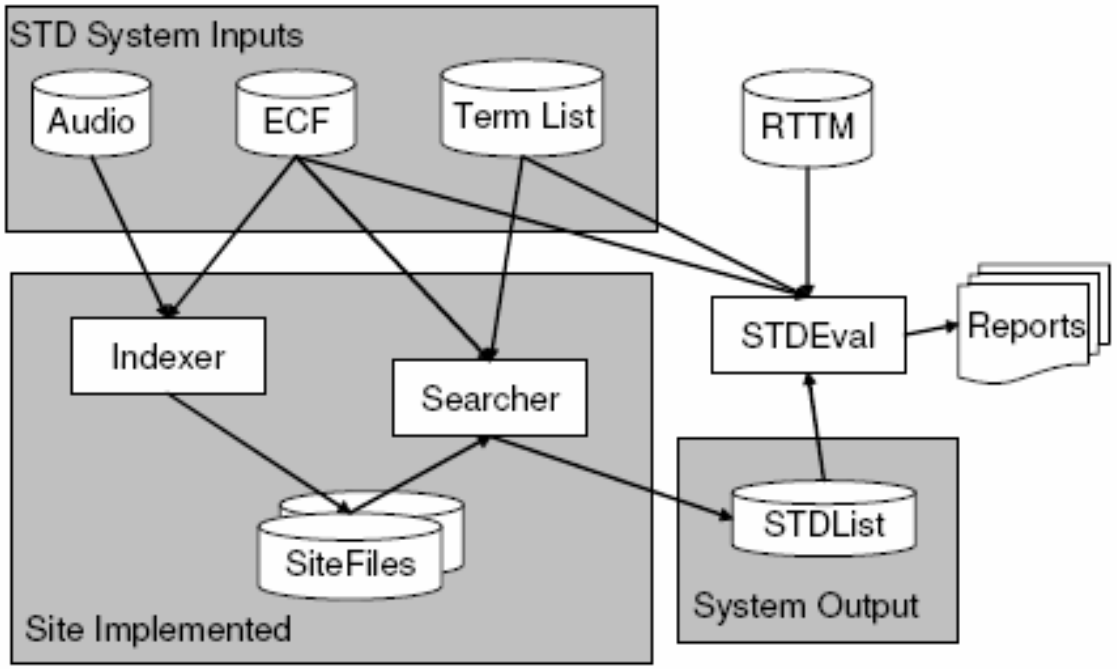


Figure 1: System and evaluation inputs and outputs

Implementation Details (2/4)

- ECF：Experiment Control Files are the mechanism the evaluation infrastructure uses to specify time regions within an audio recording, the language, and the source type specified for the experimental condition.
- TermList Files：A TermList file is an XML-formatted text file that defines the search terms to be processed by an STD system.

```
<termlist
        ecf_filename="expt_06_std_eval06_mand_all_spch_expt
        _1" version ="20060511-0900"
        language="english">
<term termid="dev06-0001">
        <termtext>find</termtext></term>
<term termid="dev06-0002">
        <termtext>many items</termtext></term>
</termlist>
```

Implementation Details (3/4)

- STDList Files：The STDList file is an XML-formatted file produced during the search phase of a system. It contains all the runtime information aswell as the search term output generated by the system.

```
<stdlist
        termlist_filename="expt_06_std_eval06_mand_all_spch_
        expt_1_Dev06.tlist.xml"
        indexing_time="74390.00"
        index_size="32959402"
        language="english"
        system_id="Phonetic subword lattice search">
<detected_termlist termid="dev06-0001"
                term_search_time="24.3" oov_term_count="0">
    <term file="NIST_20020214-1148_d05_NONE"
                channel="1" tbeg="6.956" dur="0.53"
        score="4.115" decision="YES"/>
    <term file="NIST_20020214-1148_d05_NONE"
                channel="1" tbeg="45.5" dur="0.3" score="4.65"
        decision="NO"/>
</detected_termlist>
</stdlist>
```

Implementation Details (4/4)

- RTTM：The Rich Transcription Time Mark file format (with ".rttm" filename extension) will be used to represent the reference transcription.

Rich Text object types and subtypes

Type	Subtypes
Structural types:	
SEGMENT	**eval**, or (none)
NOSCORE	(none)
NO_RT_METADATA	(none)
STT types:	
LEXEME	**lex**, **fp**, **frag**, **un-lex** , **for-lex**, **alpha** , **acronym** , **interjection** , **propernoun** , and **other**
NON-LEX	**laugh**, **breath**, **lipsmack**, **cough**, **sneeze**, and **other**

Evaluation— Detection Error Tradeoff (1/2)

- Basic detection performance will be characterized in the usual way via standard detection error tradeoff (DET) curves of miss probability (P_{Miss}) versus false alarm probability (P_{FA}).

$$P_{miss}(term,\theta) = 1 - \frac{N_{correct}(term,\theta)}{N_{true}(term)}$$

$$P_{FA}(term,\theta) = \frac{N_{spurious}(term,\theta)}{N_{NT}(term)}$$

- θ : detection threshold
- $N_{correct}(term,\theta)$: the number of correct (true) detections of *term* with a score greater than or equal to θ .
- $N_{true}(term,\theta)$: the true number of occurrences of term in the corpus.
- $N_{spurious}(term,\theta)$: the number of spurious (incorrect) detections of term with a score greater than or equal to θ .
- $N_{NT}(term,\theta)$: the number of opportunities for incorrect detection of *term* in the corpus (= “Non-Target” *term* trials).

Evaluation— Detection Error Tradeoff (2/2)

- Since there is no discrete specification of “trials”, the number of Non-Target trials for a term, $N_{NT}(term)$, will be defined somewhat arbitrarily to be proportional to the number of seconds of speech in the data under test. Specifically:

$$N_{NT}(term) = n_{tps} * T_{speech} - N_{true}(term)$$

- n_{tps} : the number of trials per second of speech (n_{tps} will be set arbitrarily to 1)
- T_{speech} : the total amount of speech in the test data (in seconds).
- P_{Miss} and P_{FA} will be computed separately for each term and then averaged over the selected terms, giving equal weight to each search term:

$$P_{Miss}(\theta) = \underset{term}{average}\{P_{Miss}(term,\theta)\}$$

$$P_{FA}(\theta) = \underset{term}{average}\{P_{FA}(term,\theta)\}$$

Evaluation— System Detection Performance

- Overall system detection performance will be measured in terms of an application model by assigning a value to each correct output and a cost (= negative value) to each incorrect output.
- Two definitions of overall system value will be used：

  - Occurrence-weighted value ($Value_0$)：

$$Value_o = \frac{\sum_{term} \{V * N_{correct}(term,\theta) - C * N_{spurious}(term,\theta)\}}{\sum_{term} \{V * N_{true}(term)\}}$$

  - Term-weighted value ($Value_T$)：

$$Value_T(\theta) = 1 - \underset{term}{average}\{P_{Miss}(term,\theta) + \beta * P_{FA}(term,\theta)\}$$

- $\beta = \frac{C}{V} * (\Pr_{term}^{-1} - 1)$
- θ ： the detection threshold
- For the current evaluation, the cost/value ratio, C/V, will be 0.1, and the prior probability of a term, $\Pr_{term}$, will be 10^{-4}.

Evaluation— System Detection Performance

- System performance will be analyzed by computing the system's value conditioned on source type and various term subsets.
- So that NIST may perform a comprehensive analysis of system performance, systems will be required to output, for each search term, more putative occurrences than those which the system determines will maximize system output Value. System output must therefore also include a binary indication of detection, determined so as to maximize system output Value. This will enable NIST to measure system performance at various levels of output (for example, at levels of 1, 10, 100 and 1000 occurrences per term) in addition to the system's determination of the optimum number of occurrences needed to maximize Value.

Evaluation — Primary Evaluation Measures

- ATWV (Actual Term-Weighted Value)：

  The ATWV is the detection value attained by the system as a result of the system output and the binary “YES/NO” decisions output for each putative occurrence.

- MTWV (Minimum Term-Weighted Value)：

  MTWV is the minimum term-weighted value found over the range of all possible values of .

Evaluation — Processing Issues

- The processing issues of speed and memory are to be reported separately for both preprocessing (“indexing”) and for search.
- The indexing time and the size of the indexing database are to be reported for each corpus (language and source type) indexed.
- The search time is to be reported for each term and for each language/source type.

Evaluation —Occurrence judging

- The system output occurrence will be judged as correct if the mid-point of the system output occurrence is less than or equal to 0.5 seconds from the time span of a known occurrence of the search term. Note, however, that mapping will be one-to-one.
- Therefore, if there are two output occurrences that are both permissible matches to only one known occurrence, then one of the output occurrences will be judged as incorrect.
- Similarly, if there are two known occurrences that are both permissible matches to only one system output, then that system output will be judged as correct for only one of the known occurrences.

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
[2]Microsoft Research Asia

Outline

- Introduction
- Lattice-based word spotting
- Lattice generation and search with word and sub-word units
- Lattice post-processing
- System combination
- Results
- Conclusions

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.
- 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”.
- However, further research 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.

Introduction

- 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.
- Combining Word-based and Phoneme-based Approach shows significant improvement .

Introduction- Characteristics of Chinese

The (Mandarin) Chinese language has several distinctive characteristics compared to English：

- Graphemically, a *word* is a sequence of *characters*. The set of most common 6000 Chinese characters has a sufficient coverage for most user scenarios. A character alone is a word itself, and has its own meaning. Thus the OOV problem of LVCSR system is significantly reduced as those characters themselves are in the dictionary as well.

Introduction- Characteristics of Chinese

- Phonetically, a *character* is a Consonant-Vowel structured *syllable*. 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. 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.

Introduction

- We examine lattice-based spoken term detection for Chinese spontaneous speech：

  - word

  - 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.

- Further improvement is achieved by lattice post-processing method and system combination.

Lattice-based Word Spotting –Lattice Definition (1/2)

- A lattice $L = (N, A, n_{start}, n_{end})$ is a directed acyclic graph (DAG).

  N : the set of nodes

  A : the set of arcs

  $n_{start}, n_{end} \in N$: the unique initial and unique final node, respectively

- Each node $n \in N$ has an associated time $t[n]$ and possibly an acoustic or language-model context condition.
- Arcs are 4-tuples $a = (S[a], E[a], I[a], w[a])$.

  $S[a], E[a] \in N$: the start and end node of the arc

  $I[a]$: the word (or sub-word) identity

  $w[a]$: a weight assigned to the arc by the recognizer.

$$w[a] = P_{ac}(a)^{\frac{1}{\lambda}} * P_{LM}(a)$$

- $P_{ac}(a)$: acoustic likelihood ; P_{LM} : LM probability ; λ : LM weight

Lattice-based Word Spotting –Lattice Definition (2/2)

- 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 $w[\pi] = \prod_{k=1}^{K} w[a_k]$.

Lattice-based Word Spotting
- Posterior Lattice Representation (1/4)

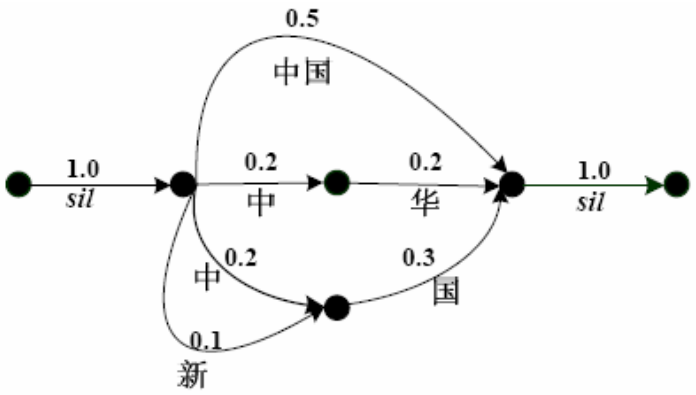
- Chelba proposed a posterior-probability based approximate representation in which word hypotheses are merged w.r.t. word position, which is treated as a hidden variable.
- It easily integrates with text search engines, as the resulting index resembles a normal text index in most aspects. However, it trades redundancy w.r.t. LM state and context for uncertainty w.r.t. word position, and only achieves a small reduction of index entries. Also, time information for individual hypotheses is lost, which we consider important for navigation and previewing.

Lattice-based Word Spotting
- Posterior Lattice Representation (2/4)

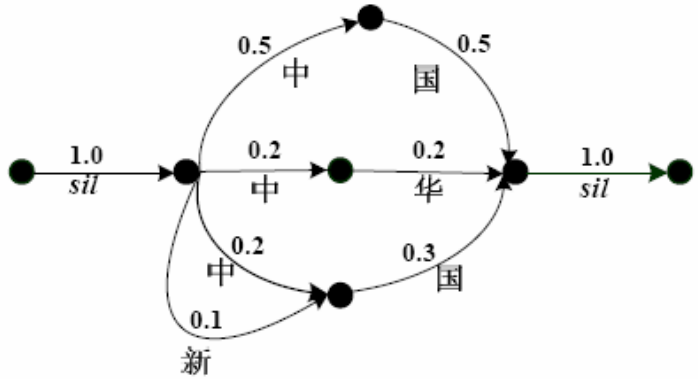
We take a three-step approach：

- First, following (Chelba, 2005), we use a posterior-probability representation, as posteriors are resilient to approximations and can be quantized with only a few bits.
- Second, we reduce the inherent redundancy of speech lattices by merging word hypotheses with same word identity and similar time boundaries, hence the name “Time-based Merging for Indexing” (TMI).
- Third, the resulting hypothesis set is represented in the index by reinterpreting existing data fields and repurposing auxiliary bits.

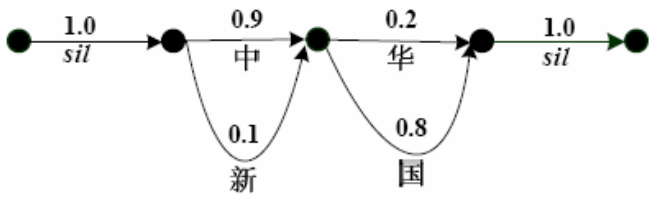
Lattice-based Word Spotting
- Posterior Lattice Representation (3/4)



(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.

$$P_{arc} = \frac{\alpha_{s[a]} * w[a] * \beta_{E[a]}}{\alpha_{n_{end}}}$$

$$P_{node}[n] = \frac{\alpha_n * \beta_n}{\alpha_{end}}$$

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

Lattice-based Word Spotting
- Posterior Lattice Representation (4/4)

- With the posterior lattice representation, the phrase posterior of query string *Q* is computed as:

$$P(*,t_s,Q,t_e,*|O)=\sum_{\substack{\pi=(a_1\cdots a_k):\\ t[S[\pi]]=t_s\wedge t[E[\pi]]=t_e\wedge I[\pi]=Q}}\frac{P_{arc}[a_1]\cdots P_{arc}[a_k]}{P_{node}[S[a_2]]\cdots P_{node}[S[a_k]]}$$

- 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.

Lattice Generation and Search with Word and Sub-word Units(1/4)

I. Lattice Generation and Matching

- 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.
- 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.

Lattice Generation and Search with Word and Sub-word Units(2/4)

II. 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 “中國人” (“Chinese People”) can be broken into either “中國人”(single word) or “中國-人” (two words). Thus an arc in lattice with “中國人” will not match query “中國” (“Chinese”) though it should.
- The problem could be solved by converting word lattices to character lattices, and matching queries by characters only.

Lattice Generation and Search with Word and Sub-word Units(3/4)

- 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

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

  - 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.

Lattice Generation and Search with Word and Sub-word Units(4/4)

III. 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.

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 , 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.

System Combination

- 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:

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

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.

Results (1/3)

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

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

Results (2/3)

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.

Results (3/3)

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

Conclusions

- This paper has examined lattice-based spontaneous 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%.

Reference

[1] NIST, "The Spoken Term Detection (STD) 2006 evaluation plan", http://nist.gov/speech/tests/std/2006/docs/std06-evalplan-v10.pdf, 2006.

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

[3] Sha Meng, Peng Yu, Frank Seide, and Jia Liu, "A Study of Lattice-Based Spoken Term Detection", ASRU, 2007.

[4] Sha Meng, Peng Yu, Jia Liu, and Frank Seide,” Fusing Multiple Systems Into A Compact Lattice Index for Chinese Spoken Term Detection”, ICASSP, 2008