APPROXIMATE WORD-LATTICE INDEXING WITH TEXT INDEXERS: TIME-ANCHORED LATTICE EXPANSION

Peng Yu, Yu Shi, and Frank Seide

Microsoft Research Asia, 5F Beijing Sigma Center, 49 Zhichun Rd., 100080 Beijing, P.R.C.

{rogeryu,yushi,fseide}@microsoft.com

ABSTRACT

We address the problem of how to represent or approximate speech lattices to be indexed with existing text indexers. We present a method named Time-Anchored Lattice Expansion (TALE), which can be implemented by a Standard Text Indexer (STI). On a 170-hour lecture set, we compare TALE with other lattice indexing methods: confusion networks, Position-Specific Posterior Lattices (PSPL), and Time-based Merging for Index (TMI). All methods achieve accuracies comparable to searching raw lattices when the corresponding index structures and phrase matching algorithms are used. However, when implemented with an STI, TALE significantly outperforms all other methods. Compared to indexing linear text, TALE improves accuracy by 30-60% for multi-word phrase searches and by 130% for two-term AND queries.

Index Terms— lattice indexing, Standard Text Indexer, keyword spotting.

1. INTRODUCTION

The tremendous progress in audio compression and storage technologies and the pervasive adoption of the Intra/Internet has fostered a dramatic increase of the use of digital media, such as online lecture videos, archived meetings or conference calls, and voicemails. Search engines to deal with digital audio or video as well as text materials become important. Currently, most existing systems rely on the anchor text, surrounding text, closed captions, and metadata of the audio or video file. However, such information is not always available, and sometimes is not discriminative at all. On the other hand, automatic speech recognition makes it possible to directly index the speech part of those digital medias.

Typical audio and video for the Internet and enterprise scenario is still a challenge for today's speech-recognition technology, which achieves word accuracies of only 50-70% [1, 2, 3]. To maximize search accuracy, the probabilistic nature of speech recognition must be considered [4]. A significant improvement can be achieved through incorporating word confidence scores and alternative recognition candidates by searching *word lattices* instead of linear speech-to-text output [5, 6, 7, 8]. Word lattices are a compact representation of word candidates and their scores and time information.

On the other hand, there is a request to reuse existing text search engines, because search engines are complex systems involving substantial investments in both development and deployment, and also in order to seamlessly search text and audio/video materials. This poses two challenges: (a) raw word lattices can be as large as 100 times the size of text or more, which is beyond the range typical text search engines are optimized for; and (b) a Standard Text Indexer (STI) assigns a unique word position to each word – thus it cannot represent alternates with different time boundaries or spanning multiple words, and phrase queries match words in consecutive positions – thus the link information in lattices will not be properly used.

In our previous work [7], we have proposed the method named Time-based Merging for Index (TMI), which significantly reduces the lattice size while still maintains the search accuracy. However, TMI lattices contain words spanning multiple positions, which is not compatible with the STI index structure.

Mangu in [9] proposed to convert lattices to "confusion networks" – also known as "sausages," for Minimal Word Error Rate decoding. The algorithm was used in [10, 11] for speech indexing. Confusion networks align words to word positions, therefore are compatible with the STI index structure. However, the null links in confusion networks require special consideration for the phrase matcher. Besides, it is very inefficient to inverse index and match those null links due to the huge occurrence list.

In [6] Chelba proposed Position Specific Posterior Lattices (PSPL). To be compatible with the STI, PSPL enumerate all the paths and index the word sequence on each path like a text. This results in a significant amount of position over-generation for each word, and becomes infeasible for more than a few sentences, thus chopping of the audio is required. Besides, PSPL cannot represent time information – which is useful for audio browsing.

In [8] we proposed the method named Time-Anchored Lattice Expansion (TALE), and showed that TALE can also be represented by an STI while maintaining the accuracy of raw lattices and keeping the index size relatively small. In this paper, we compare TALE with other indexing methods: TMI, PSPL, confusion network, and simple binning. Our results show that when implementing lattice indexing by an STI, TALE significantly outperforms all other methods.

This paper is organized as follows. In section 2, we review prior work of lattice based speech indexing, and section 3 discusses how to approximate lattices for use with text indexers. Section 4 introduces the TALE method, section 5 presents experimental results, and section 6 concludes the paper.

2. LATTICE-BASED KEYWORD SPOTTING

We first recapitulate previous work on lattice-based search. A word lattice $\mathcal{L} = (\mathcal{N}, \mathcal{A}, n_{\text{enter}}, n_{\text{exit}})$ is a weighted directed acyclic graph (DAG) where arcs $\mathcal{A}$ represent word hypotheses with recognizer scores, and nodes $\mathcal{N}$ the connections between them, encoding times and possibly context conditions. [1] n_{enter} and $n_{\text{exit}} \in \mathcal{N}$ are the unique initial and final node, respectively. The recognizer score of a word hypothesis is used as the arc weight:

$$q_{n_s,w,n_e} \quad = \quad p^{\frac{1}{\lambda}}(O(t_{n_s}...t_{n_e})|n_s, w, n_e) \cdot P(w|n_s),$$

where $p(O(t_{n_s}...t_{n_e})|n_s, w, n_e)$ is the likelihood for acoustic observation $O(t_{n_s}...t_{n_e})$ given word w, its time boundaries (t_s, t_e),

[1] Alternative definitions of lattices are possible, e.g. nodes representing words and arcs representing word transitions.

and its cross-word triphone context (n_s, n_e). $P(w|n_s)$ is the language-model (LM) probability of word w to follow its LM history (encoded in n_s). λ is the well-known LM weight.[2]Consider $q_{n_s,w,n_e} = 0$ for non-existent arcs.

The lattice representation allows to answer one question of interest: Given our observed audio recording O, what is the probability $P(*\text{-}t_s\text{-}w\text{-}t_e\text{-}*|O)$ that a particular word w was spoken at a particular time $t_s...t_e$? This quantity is called the *word posterior probability*. Despite its name, it is defined over *paths*, and $*\text{-}t_s\text{-}w\text{-}t_e\text{-}*$ shall denote the set of paths that contain w with boundaries t_s and t_e. To compute it, we sum over all nodes (n_s, n_e) with the given time points (t_s, t_e):[3]

$$P(*\text{-}t_s\text{-}w\text{-}t_e\text{-}*|O) = \sum_{\substack{(n_s,n_e):\\ t_{n_s}=t_s \wedge t_{n_e}=t_e}} P(*\text{-}n_s\text{-}w\text{-}n_e\text{-}*|O),$$

where the *arc posterior* $P(*\text{-}n_s\text{-}w\text{-}n_e\text{-}*|O)$ is computed as:

$$P(*\text{-}n_s\text{-}w\text{-}n_e\text{-}*|O) = \frac{\alpha_{n_s} \cdot q_{n_s,w,n_e} \cdot \beta_{n_e}}{\beta_{n_{\text{enter}}}}$$

and the *forward probability* α_{n_s} and *backward probability* β_{n_e} represent the sum over all paths from sentence start n_{enter} to n_s and n_e to sentence end n_{exit}, respectively. They can be computed conveniently with the forward-backward recursion [12]. $\beta_{n_{\text{enter}}}$ is the total probability over all paths.

Relevance-ranking formulas often use the *term frequency* TF_w (per-document keyword occurrence). Its expected value can be computed from the lattice as:

$$\begin{aligned} E_{w|O}\{\text{TF}_w\} &= \sum_{\substack{\forall m,n_0...n_m:\\ n_0=n_{\text{enter}} \wedge\\ n_m=n_{\text{exit}}}} P(n_0\text{-}w_1\text{-}n_1\text{-}...\text{-}w_m\text{-}n_m) \cdot \sum_{\forall i: w_i=w} 1 \\ &= \sum_{\forall n,n'} P(*\text{-}n\text{-}w\text{-}n'\text{-}*|O) \end{aligned}$$

One would also want to answer the same question for multi-word sequences $(w_1 w_2 ... w_m)$, not only to support explicitly quoted phrase queries, but also because sequence matches are significantly more accurate, and query terms often occur in sequence (implicit phrases). The *phrase posterior* $P(*\text{-}t_s\text{-}w_1...w_m\text{-}t_e\text{-}*|O)$ can be computed by summing over all m-arc paths with the given boundaries t_s and t_e:

$$P(*\text{-}t_s\text{-}w_1...w_m\text{-}t_e\text{-}*|O) = \sum_{\substack{\forall m,n_0...n_m:\\ t_{n_0}=t_s \wedge\\ t_{n_m}=t_e}} P(*\text{-}n_0\text{-}w_1\text{-}n_1\text{-}...\text{-}w_m\text{-}n_m\text{-}*|O)$$

$$P(*\text{-}n_0\text{-}w_1\text{-}n_1\text{-}...\text{-}w_m\text{-}n_m\text{-}*|O) = \frac{\alpha_{n_0} \prod_{i=1}^{m} q_{n_{i-1},w_i,n_i} \beta_{n_m}}{\beta_{n_{\text{enter}}}}$$

In this paper, we actually use an equivalent, more convenient representation, which we call the *posterior lattice*. In a posterior lattice, arc weights are not q_{n_s,w,n_e} but directly the pre-computed arc posteriors $P(*\text{-}n_s\text{-}w\text{-}n_e\text{-}*|O)$. This representation still allows exact computation of phrase posteriors:

$$P(*\text{-}n_0\text{-}w_1\text{-}n_1\text{-}...\text{-}w_m\text{-}n_m\text{-}*|O) = \frac{\prod_{i=1}^{m} P(*\text{-}n_{i-1}\text{-}w_i\text{-}n_i\text{-}*|O)}{\prod_{i=1}^{m-1} P(*\text{-}n_i\text{-}*|O)}$$

$$\text{with} \quad P(*\text{-}n\text{-}*|O) = \frac{\alpha_n \beta_n}{\beta_{n_{\text{enter}}}} = \sum_{\forall n'} \sum_{\forall w} P(*\text{-}n\text{-}w\text{-}n'\text{-}*|O)$$

We call the new term $P(*\text{-}n\text{-}*|O)$ *node posterior*[4]. The primary advantage of the posterior representation is that posteriors are resilient to approximations like aggressive quantization and merging of alternates with non-identical time boundaries, and they allow comparing arcs with different time durations and temporal splitting e.g. compound words. Further, the node posteriors turn out to be uncritical and can be replaced by a constant in our scenario.

3. APPROXIMATING LATTICES FOR USE WITH STANDARD TEXT INDEXERS

This section discusses the problem of how to represent or approximate word lattices such that they can be indexed with an STI.

To implement lattice indexing with an STI, words must be aligned to word positions, forming a sausage-like lattice. Word posteriors have to be stored e.g. as part of the supplementary "ranking information" and must be used by the ranker. Node times are needed only for the result display, and are not stored in the inverted index.

To have the ranker use posteriors can be achieved by a trick. Typical indexer designs use the "ranking information" as an abstract type index into a weight table [13]. To use posteriors in ranking, we'd just need to change the weight table accordingly. Thus, the remaining issue is phrase matching.

An STI phrase matcher requires words belonging to phrases to be in consecutive word positions, i.e. some words must be aligned to multiple slots (over-generation). Obviously, excessive over-generation leads to high false alarm rate, and large index size. We have to set priorities.

We propose the following criterions for the task: (a) retain the expected term frequencies $E_{w|O}\{\text{TF}_w\}$ (they matter for ranking); (b) keep time points of individual hits accurate enough to allow playback; (c) have all phrases *up to three words* in consecutive word positions; (d) keep the index size reasonable. The next section will introduce our solution.

4. TIME-ANCHORED LATTICE EXPANSION

First, we choose the time boundaries of the best path as "anchor points" $t_0...t_T$, and align each node n to the closest anchor point t_i, denoted by $i = s_n$. Each word is also aligned to a word position by its starting node. We call this "binning."

Binning itself does not keep phrases in consecutive word positions. To realize that, we introduce the "Δ_δ-Expansion".

Define the conditional probability that word w happens as the δ-th path token after a given node n:

$$P(w|n,\delta,O) = \frac{\sum_{\substack{\forall n_i,w_i:\\ i=1...\delta \wedge w_\delta=w}} P(*\text{-}n\text{-}w_1\text{-}n_1\text{-}...\text{-}w\text{-}n_\delta\text{-}*|O)}{P(*\text{-}n\text{-}*|O)},$$

"Δ_δ-Expansion" computes the probability distribution for slot i of word w as δ-th token in phrase:

$$P_\delta(w|i,O) = \sum_{\forall n: s_n+\delta=i} P(*\text{-}n\text{-}*|O) \cdot P(w|n,\delta,O).$$

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

[3]In most applications, one would also relax the time boundaries, e.g. extending the sum to include alternate boundaries with significant overlap.

[4]In [5], a similar concept is implemented by weight pushing.

It is easy to see that $E_{w|O}\{\mathrm{TF}_w\}$ remains unchanged for all w. Obviously, binning is equivalent to Δ_0-Expansion.

To guarantee to retain all M-word phrases in consecutive slots, we interpolate multiple Δ_δ-Expansions:

$$P(w|i,O) \quad = \quad \sum_{\delta=0}^{M-1} \alpha_\delta \cdot P_\delta(w|i,O),$$

with $\sum \alpha_\delta = 1$. It can be shown that any M-word phrases will appear in consecutive slots with this merging. δ can also be negative, which means expanding words left to the given node. We actually use Δ_{-1}, Δ_0 and Δ_1 in our experiments to keep all the trigrams. The weights α_δ would ideally be optimized on a development set to maximize the overall accuracy, but it is not necessary: Experiments show that using equal weights yields almost as good result as full lattices. The time information is retained by the anchor points. We name this method Time-Anchored Lattice Expansion (TALE).

5. EXPERIMENTAL RESULTS

5.1. Setup

In [8], we have evaluated TALE on four production-size corpora: lectures, conversations, meetings, and voicemails, totaling 600 hours. In this paper, due to limited space, we only report results on the lecture set, but the conclusion is consistent across other sets.

The lecture set – MIT iCampus lectures [3] – is 170 hours long with 169 lectures. The whole audio was pre-segmented into 65927 sentences. Raw lattices were generated with a speaker-independent LVCSR system. The acoustic model was trained on the 1700-hour Switchboard "Fisher" telephone-speech set [2]. Due to limited LM data for lectures, we partitioned the test set into 10 parts, and recognized each part with an LM trained on the transcripts of the remaining 9 parts, keeping training and test disjunct. The Word Error Rate (WER) for the test set is 46.6%.

We evaluate our method on keyword search without relevance weighting. We include multi-word phrase queries, single-word queries, and two-term AND queries (each term can be single or multi-word). The keyword set is synthetic and consists of noun phrases chosen from the transcripts such that for each query there are at most two matching documents. Three accuracy metrics are used:

- FOM: The NIST Figure Of Merit defined as the detection/false-alarm curve averaged over [0..10] false alarms per keyword per h hours. Instead of the original $h = 1$, we use h=data set duration. The time tolerance range for a match is extended to the whole sentence, so that PSPL (which has no time information for individual words) can also be evaluated;
- mAP: mean average precision, ranking documents by the probability that the single or multi-word query Q occurs in the document at least once:

$$P(Q \text{ in } doc) \quad = \quad 1 - \prod_{\forall \text{hits } h \text{ for } Q} (1 - P(h|doc)) \quad (1)$$

  and for AND queries:

$$P(X \text{ AND } Y \text{ in } doc) = P(X \text{ in } doc) \cdot P(Y \text{ in } doc) \quad (2)$$

- R_{75}/R_{50}: document Recall at Precision 75%; and at 50% for single-word queries, for which it proved difficult to get enough observation points above 75%.

Both mAP and R_{75} rely on a joint ranking among all query terms. Therefore we normalize the word and phrase posteriors using three 2-state Hidden-State Maximum-Entropy models [14] for single words, two-word phrases, and longer phrases, respectively. This improves AND-query results by up to 5 points.

5.2. Speech-To-Text Transcript vs. Lattice Indexing

In Table 1, the first and second rows compare accuracies for the simplest approach, indexing speech-to-text (STT) plain-text transcripts, with raw lattice indexing. To be able to compare transcript results with lattices, we attached posteriors from the lattice to each transcript word. (Without that, there is only one Precision/Recall point, for which the results are very similar to the R_P result.)

The first three result columns show accuracies for phrase queries. From STT transcripts to raw lattice, a significant improvement is observed. For FOM and mAP, relative improvements are 63 and 57%, respectively, and 28% for R_{75}. For the single-word queries (next three columns), improvements for FOM and mAP are in the 30% range, and none for R_{50}. The next three columns show AND-query results. Both mAP and R_{75} increase by a solid 2.4 times.

5.3. Lattice Indexing Methods

In the next experiment, we compare different lattice indexing methods, which is shown in the second block of Table 1. Corresponding index structure and phrase matching algorithm are used for each method. PSPL and TALE use STI directly. TMI has an index keeping extent for each word, which are used in phrase matching. Confusion networks have null links skipable. We also compare the binning method introduced in section 4. As binning breaks phrases, a special phrase matcher called *D1 matching* is used , which allows consecutive words to be in same slot or with one extra slot in between (*D1* here for distance-one).

The table shows that, for FOM and mAP, which are principally dominated by recall, all methods have comparable performance with raw lattices. Interestingly, for phrase and AND queries, indexed lattices are even better than raw lattices, because the indexing methods introduce (incorrect) extra paths, which results in better recalls.

For R_{75}/R_{50}, which emphasizes on precision, we observe significant degradation for PSPL. This is because PSPL distribute a single word probability into too many slots, thus the probabilities calculated in Eq. 1, 2 are seriously distorted. (Note that for PSPL, the ETFs of words are retained as well). There is also a smaller decrease on phrase and AND queries for binning, which is caused by the over-tolerance from D1 matching.

The last column shows the index size. Binning and confusion networks have the smallest size down to 20, while PSPL have as high as 329 due to excessive over-generation.

5.4. Index Pruning

In the third experiment, we prune index sizes to about 10 index entries per spoken word for each method. For the confusion networks, the pruning is applied to the raw lattices with respect to likelihood scores, while for the others the pruning is on the posterior scores after lattice conversion.

For FOM and mAP, though we observe 2-3 points decrease for phrase queries and 4-6 points for AND queries with most methods, the accuracies are still comparable to raw lattices, excepte for PSPL, which is about 5 points worse than raw lattices for phrase queries and 10 points worse for AND queries, due to a much larger size reduction. For R_{75}/R_{50}, the degradations are smaller (in some condition even better), showing that the pruning has less impact on precision.

Table 1. Search results for phrase queries, single-word queries, and two-term queries of the form X AND Y where X and Y may be phrases. Shown are Figure of Merit (FOM) per keyword hit, per-document mean average precision (mAP), and per-document Recall at a certain Precision cut-off (R_P) for P=75% and 50% for multi and single-word queries, respectively. "Index size" is index entries per spoken word.

query type:	phrase queries			single-word queries			AND queries		index size	
configuration	FOM	mAP	R_{75}	FOM	mAP	R_{50}	mAP	R_{75}	edg/wd	nd/wd
STT transcript with confidence	40.3	42.7	43.4	36.0	44.2	45.2	26.1	26.1	1.0	1.1
raw lattice	65.6	67.2	55.7	48.2	55.9	45.4	63.3	61.6	1617	239.6
PSPL	67.6	70.5	47.7	48.2	53.8	27.6	69.6	59.7	329.0	2.5
TMI	67.9	69.6	58.0	48.2	55.9	45.4	66.3	63.9	46.3	2.9
confusion network *	67.2	69.8	57.1	47.5	55.4	45.0	66.6	61.6	23.8	3.7
binning	67.2	69.2	52.8	48.2	55.9	45.4	67.2	62.5	20.1	1.1
TALE (binning with expansion)	68.0	70.2	57.9	48.2	55.5	44.7	67.8	63.7	36.9	1.2
pruning to 10 edges per word										
PSPL	60.6	63.4	50.6	47.2	53.1	42.1	53.8	54.3	12.4	2.5
TMI	65.1	67.1	58.3	48.0	55.4	45.4	60.4	61.0	10.0	1.7
confusion network	65.7	68.0	57.7	48.0	55.5	45.5	62.3	61.9	9.4	1.9
binning	64.8	66.9	53.1	48.1	55.5	45.4	61.5	61.3	7.9	1.1
TALE (binning with expansion)	65.1	67.5	57.9	48.0	55.1	44.8	61.6	61.7	11.8	1.2
pruning to 10 edges per word, with standard text index structure and standard phrase matcher										
PSPL	60.6	63.4	50.6	47.2	53.1	42.1	53.8	54.3	12.4	2.5
confusion network	26.7	28.4	25.1	48.0	55.5	45.5	17.9	17.8	9.4	1.9
binning	61.3	63.6	56.9	48.1	55.5	45.4	56.1	56.5	7.9	1.1
TALE (binning with expansion)	65.1	67.5	57.9	48.0	55.1	44.8	61.6	61.7	11.8	1.2

* About 1% segments failed due to huge memory complexity.

5.5. Implemented with Standard Text Indexers

In the last experiment, we implement lattice indexing with an STI, which is the primary goal of this paper. Obviously only phrase queries and AND queries will be affected. TMI is excluded in this experiment as the index structure is not compatible.

As the last block of the table shows, confusion networks' performance drops down dramatically because null links are not considered by the STI phrase matcher – all metrics go down below 30% for phrase queries, and below 20% for AND queries. For binning, about 4-point accuracy drops are observed on most metrics due to not using D1 matching. Results for PSPL and TALE are not changed. Compared with other methods, TALE are significantly better.

6. CONCLUSION

We addressed the problem of how to represent or approximate word lattices such that they can be indexed with existing text indexers. We proposed a method named TALE (Time-Anchored Lattice Expansion), which (a) retains rough time information for audio browsing; (b) keeps expected term frequency for each word (they matter for ranking); (c) is guaranteed to keep phrases up to 3 words in consecutive slots (thus a text phrase-matcher can find them); (d) keeps index size reasonable.

On a 170-hour lecture set, we compare TALE with other lattice indexing methods: PSPL, TMI, confusion network, and binning. With corresponding index structures and phrase matchers, all methods achieve accuracies comparable to searching raw lattices (except that PSPL were 5-10 points worse for phrase and AND queries when pruned to 10 index entries per spoken word). But when implemented with a Standard Text Indexer, TALE are significantly better than all other methods.

Compared to indexing linear text, the presented TALE method can achieve an accuracy improvements of 30-60% for multi-word phrase searches, and of 130% for two-term AND queries, which rely more on recall. Most importantly, this improvement can be achieved with a Standard Text Indexer.

7. REFERENCES

[1] M. Padmanabhan, G. Saon, J. Huang, B. Kingsbury, and L. Mangu, Automatic Speech Recognition Performance on a Voicemail Transcription Task. *IEEE Trans. on Speech and Audio Processing, Vol. 10, No. 7, 2002*.

[2] G. Evermann, H. Y. Chan, M. J. F. Gales, B. Jia, X. Liu, D. Mrva, K. C. Sim, L. Wang, P. C. Woodland, K. Yu, Development of the 2004 CU-HTK English CTS Systems Using More Than Two Thousand Hours of Data. *Proc. Fall 2004 Rich Transcription Workshop (RT-04), 2004*.

[3] J. Glass, T. J. Hazen, L. Hetherington, C. Wang, Analysis and Processing of Lecture Audio data: Preliminary investigation. *Proc. HLT-NAACL'2004 Workshop: Interdisciplinary Approaches to Speech Indexing and Retrieval*, Boston, 2004.

[4] P. Yu, K. J. Chen, C. Y. Ma, F. Seide, Vocabulary-Independent Indexing of Spontaneous Speech, IEEE Transactions on Speech and Audio Processing, Vol.13, No.5.

[5] M. Saraclar, R. Sproat, Lattice-based search for spoken utterance retrieval. *Proc. HLT'2004*, Boston, 2004.

[6] C. Chelba and A. Acero, Position specific posterior lattices for indexing speech. *Proc. ACL'2005*, Ann Arbor, 2005.

[7] 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'06*, New York, 2006.

[8] F. Seide, P. Yu, Y. Shi, Towards Spoken-Document Retrieval for the Enterprise: Approximate Word-Lattice Indexing with Text Indexers, *to appear in Proc. ASRU'2007*, Kyoto, 2007.

[9] L. Mangu, E. Brill, A. Stolcke, Finding Consensus in Speech Recognition: Word Error Minimization and Other Applications of Confusion Networks. *Computer, Speech and Language, 14(4):373-400*.

[10] T. Hori, I. L. Hetherington, T. J. Hazen, and J. R. Glass, Open-Vocabulary Spoken Utterance Retrieval Using Confusion Networks, *Proc. ICASSP'2007*, Honolulu, 2007.

[11] J. Shao, Q. W. Zhao, P. Y. Zhang, Z. J. Liu, Y. H. Yan, A Fast Fuzzy Keyword Spotting Algorithm Based on Syllable Confusion Network, *Proc. InterSpeech'2007*, Antwerp, 2007.

[12] F. Wessel, R. Schlüter, and H. Ney, Using posterior word probabilities for improved speech recognition. *Proc. ICASSP'2000*, Istanbul, 2000.

[13] S. Brin and L. Page, The anatomy of a large-scale hypertextual Web search engine. *Computer Networks and ISDN Systems, 30(1-7):107-117*.

[14] P. Yu, J. Xu, G. L. Zhang, Y. C. Chang, and F. Seide, A Hidden-State Maximum Entropy Model for Word Confidence Estimation. *Proc. ICASSP'2007*, Honolulu, 2007.