Spoken Term Detection Based on Improved Index Structure

Zhen Zhang, Ji Xu, Yujing Si, Qingwei Zhao, Yonghong Yan
Key Laboratory of Speech Acoustics and Content Understanding, Institute of Acoustics,
Chinese Academy of Sciences, Beijing, 100190, China
Email: {zhangzhen, xuji, siyujing, zhaoqingwei, yanyonghong}@hccl.ioa.ac.cn

Abstract— **The performance of keyword spotting system suffers severe degradation when the index stage is so fast that the lattice may lose lots of information to retrieve the spoken terms. In this paper, we focus on this problem and present two algorithm: the first one called unconstraint word graph expansion (UWGE) and the other called dynamic position specific posterior lattice(D-PSPL). The motivation of these methods is to keep the pruned hypotheses which are discarded in the decoding procedure but may contain correct hypotheses. The proposed approaches is to eliminate the N-gram language model state limitation of lattice and reconstruct lattice to unconstrained word graph. On two Mandarin conversation telephone speech sets, we compare performance using the two methods with that on traditional trigram lattice, and our approaches give satisfying performance gains over trigram lattice. The experiment results also show that the D-PSPL algorithm is better than the UWGE algorithm in high score area.**

Index Terms— **Spoken Term Detection, Unconstraint Word Graph Expansion, Dynamic Specific Position Posterior Lattice, N-gram Lattice, Lattice Limitation**

I. INTRODUCTION

THE ever growing volume of recorded speech data collected from telephones, cell phones and internet conversation etc, poses great challenge for the spoken language processing technologies. Keyword spotting is a very important branch of speech recognition, which is the task of detecting the occurrences of predefined keywords in the unconstrained audio stream.

The existing work done in keyword spotting can be categorized under three major approaches. The first approach is acoustic keyword spotting approach. In this approach, all words other than the keywords assumed to be garbage and are represented by garbage models. The second approach is Large Vocabulary Continuous Speech Recognition (LVCSR) approach. This approach requires complete decoding of speech signal and it outputs a completely decoded sentence [1]. The third approach of keyword spotting is a state-of-the-art approach making use of lattice (word graph) which contains alternate candidates of the decoding result. Keyword spotting uses the search in lattice and outputs whether a keyword is present in a signal or not. In this paper, we use syllable lattice in our system to search the spoken terms which has the high recall rate of the hypotheses than the word lattice [2].

Facing the challenge of huge mount of data, the keyword spotting system must be able to access the audio as fast as possible. However, the performance of the system severely suffers from a very high missing rate which leads to a serious performance degradation when the system speed is tuned to so fast as 0.36xReal-Time (RT) or more. In this paper, we consider the problem of the recall rate and propose two approaches the first algorithm named unconstrained word graph expansion (UWGE) and the other called dynamic position specific posterior lattice(D-PSPL). The motivation of these two methods is to rebuild the N-gram lattice into another form: unconstrained word graph or dynamic specific position posterior lattice. We eliminate the language model limitation of N-gram lattice and can retain most of the hypotheses generated in the decoding procedure, some of which may be pruned owing to the inherent limitations of the N-gram lattice generation algorithm. Our experiment results show that there are improvements in both figure of merit (FOM) score and equal error rate (EER) score.

The rest of the paper is organized as follows: In Section 2, we will discuss the motivation of this work; In Section 3, we will introduce the system architecture; In Section 4, we discuss the lattice generation algorithm and baseline keyword spotting paradigm; Section 5 describes the limitation of the N-gram lattice and UWGE method in detail; Section 6 describes the D-pspl method in detail; Experimental results are presented in Section 7, followed by conclusions in Section 8.

II. THE MOTIVATION OF THIS WORK

The Relationship of the Decoder Speed And the Performance

In the actual speech retrieval applications, we often need to handle vast amounts of multimedia data, so the system has high speed requirements. However, we adjust the beam width of the recoginition decoder to accelerate the speed of index, when the speed is tuned to faster the performance of the system will decline rapidly,. The reason for this is that the number of hypotheses will decrease when the speed of decoder is tuned fast. And the recall rate will also decrase which leads to the performance degradation.

A. The Hypotheses in the Word Graph

In the searching space of the recogntion decoder, all the hypotheses have existed. When the decoder execute

doi:10.4304/jsw.8.11.2807-2814

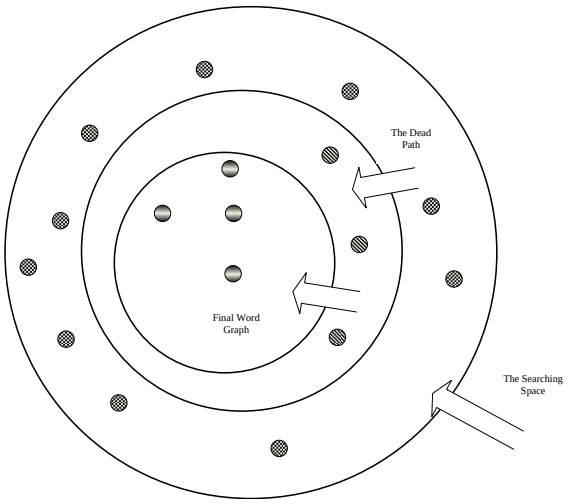


Figure 1. The Searching Space and The Activated Hypotheses

the decoding procedure, the token passing algorithm will activate a parts of the hypotheses, and these hypotheses are connected to form the path. There is prune strategy in the decoding procedure to ensure the speed of decoding. Some path will be erase from the token stack, and the paths will be removed from the word graph. However, the removed paths may contain correct hypotheses. When the searching beam is big enough, the word graph may contain more paths, as a result, it may contain more correct hypotheses. On the contrary, when the searching beam is tuned small to improve the speed of the decoder, many paths will be removed during the decoding procedure and the final word graph may contain less correct hypothess leading to the degradation of the recall rate.

As shown in Fig 1, the biggest circle represents the whole searching space in which all hypotheses exist. During the decoding procedure, some hypotheses are activated as shown in the medium circle. After the prune procedure is executed, some hypotheses is romoved from the final graph as shown in the smallest circle. In the next section, we will discuss the reason for this.

III. System Architecture

Fig. 2 shows the overall architecture of our system for Mandarin spoken term detection. There are two stages in the system:

At the index stage, the audio is fed into a large vocabulary continuous speech recognizer (LVCSR) [3], which outputs syllable lattices and we convert the lattices to index [4].

At the search stage, all the query terms are turned into syllable form, and hits of all query terms are retrieved from the inverted index. The ranker computes confidence measurement scores, and a result presentation module creates output result list with every hit's position and score.

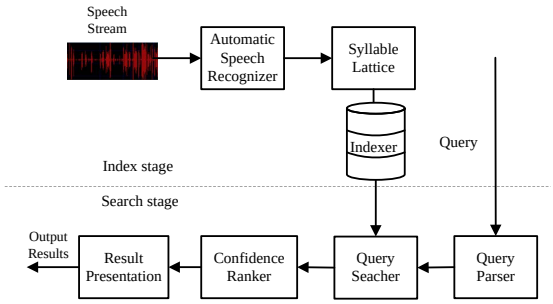


Figure 2. The Searching Space and The Activated Hypotheses

IV. Review of The Lattice Based Keyword Spotting

In this section, we will introduce the lattice generation algorithm and the base idea of the lattice based keyword spotting.

A. Lattice generation

The purpose of lattice indexing is to retain alternative candidates that the recognizer also considered, with their associated probabilities. As the production of Viterbi search of a recognizer, a lattice is a is a weighted directed acyclic graph (DAG) [5]. It is defined as $G = \{\mathcal{V}, \mathcal{E}, \mathcal{W}, \mathcal{L}, v_{start}, v_{end}\}$ where arcs $\mathcal{E}$ represent the syllable hypotheses with recognizer weight $\mathcal{W}$ and ID $\mathcal{L}$, and nodes $\mathcal{V}$ are the connections between them, encoding times and N-gram language model state. v_{start} and $v_{end} \in \mathcal{V}$ are the unique initial and final node [6], respectively.

During the decoding procedure, each hypothesis is an N+1 tuple $(u, v, .., s; t)$ which means the current s is a hypothesis ended at time t and its N-1 hypotheses' N-gram history is $(u, v...)$ [6]. The lattice records the pair of each hypothesis' time boundary and language model state. For every pair, the lattice will create a node to represent the time and language state and create a arc for the hypothesis defined on this pair. Then the lattice connects the arc to its start and end node. To remain the lattice's graph structure, the dead paths will be erased from the lattice [7].

B. Lattice based keyword spotting

Given a query Q we decompose it into a sequence of syllable units,$\{s_j, j = 1, 2, ...Q\}$. Thus we search the N-gram $\{s_1, s_2, ...s_Q\}$ in the lattice, and calculate the confidence measurement of the N-gram [8]. Given the syllable s which has the start node v_s and end node v_e, the recognizer score of a hypothesis is used as the arc weigh:

$$q_{v_s,s,v_e} = p^{\frac{1}{\lambda}}(O(t_{v_s}...t_{v_e})|v_s, s, v_e).P(s|v_s) \quad (1)$$

where $p(O(t_{v_s}...t_{v_e})|v_s, s, v_e)$ is the likelihood for acoustic observation $O(t_{v_s}...t_{v_e})$ given hypothesis s, its time

boundary (t_s, t_e), and its cross-word triphone context (v_s, v_e). $P(s|v_s)$ is the language-model (LM) probability of the hypothesis s to follow its LM history (encoded in v_s). λ is the LM weight which is used to adjust acoustic likelihood and LM probability. When we search for the spoken terms, we use word posterior probability to represent the confidence measurement of the occurrences [9]. It is defined over paths, and $*$-t_s-s-t_e-$*$ denotes the set of paths which contain s with boundaries t_s and t_e. To compute it, we sum all nodes (v_s, v_e) with given time points (t_s, t_e):

$$P(*\text{-}t_s\text{-}s\text{-}t_e\text{-}*|O) = \sum_{\substack{(v_s,v_e):\\ t_{v_s}=t_s \wedge t_{v_e}=t_e}} P(*\text{-}v_s\text{-}s\text{-}v_e\text{-}*|O) \quad (2)$$

Where the arc posterior $P(*\text{-}v_s\text{-}s\text{-}v_e\text{-}*|O)$ is computed as:

$$P(*\text{-}v_s\text{-}s\text{-}v_e\text{-}*|O) = \frac{\alpha_{v_s} \cdot q_{v_s,s,v_e} \cdot \beta_{v_e}}{\beta_{enter}} \quad (3)$$

and the forward probability α_{v_s} and backward probability β_{v_e} represent the sum over all paths from sentence start v_{enter} to v_s and v_e to sentence end v_{exit}, respectively. They can be computed conveniently with the forward-backward recursion [10]. $\beta_{v_{enter}}$ is the total probability over all paths [11]. So, the probability of the spoken terms can be computed as:

$$P(Q|O) = \sum_{s_i \in Term} P(*\text{-}v_{s_i}\text{-}s_i\text{-}v_{e_i}\text{-}*) \quad (4)$$

Where the Q's probability $P(Q|O)$ is computed by summing over m-arc paths with the given time boundaries t_s and t_e [2].

V. Unconditioned Word Graph Extension Algirithm

In this section, we will discuss the limitation of the N-gram lattice generation algorithm and propose our method to overcome the limitation.

A. The limitation of N-gram lattice

In the N-gram lattice, there must be at least one path starting from the initial node v_{enter} and ending at the exit node v_{exit} for each node in the lattice. This property of graph reachability ensures that the lattice is a fine graph structure which could be implemented by the forward-backward algorithm [12]. However, on the other hand, it makes the lattice unable to preserve the dead paths which means the paths can not not be expanded. There will be no paths passing through from the last node v_e of the dead path to the end node v_{exit} if the dead path stays in the lattice, so they are cleaned from the lattice with the risk of information loss.

Meanwhile, due to the limitation of the N-gram model , each hypothesis is an N+1 tuple $(u, v, .., s; t)$. When the pruning happens, this N+1 tuple will be pruned from the lattice. However, the influence of path prune is a bit too long, and it affects not only the last hypothesis but also a series of hypotheses before it. As a result, plenty of hypotheses are pruned just because of their dead successor arcs in the N-gram lattice.

When the speed of system is slow, the N-gram lattice could be large enough to keep the information for spoken term detection, but when the speed is fast there may be few paths kept in the structure and the performance is near to that of the STT (speech-to-text) script with confidence measurement produced by the decoder directly [13].

B. The unconstrained word graph expansion algorithm

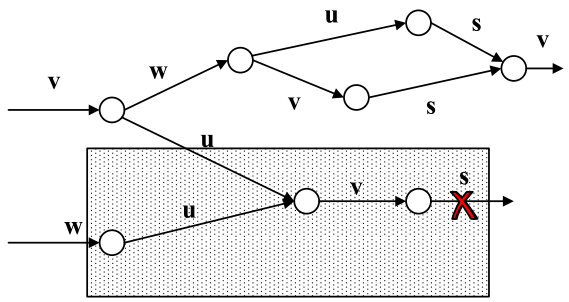
According to the limitation of N-gram lattice, we hope to keep most information from the decoding procedure by adjusting the lattice generation algorithm. The purpose is to ensure that here will be a short influence when the path pruning occurs, and the key is to eliminate the limitation of the N-gram language model.

The proposed UWGE algorithm is the approach to eliminate language model state on the nodes, which means $\forall v \in \mathcal{V}$ encodes time only. This approach is used only in index stage to change the structure of lattice and does not affect the search stage. We connect the arcs with same time boundaries to the same nodes and do not consider the history information, so when path pruning happens, the arcs before the pruned hypothesis will not be pruned due to the independent relationship with the hypothesis. The difference between constrained word graph and N-gram lattice is only the definition of the node, and the unconstrained word graph remains a weighted directed acyclic graph (DAG) and suitable for forward-backward algorithm to compute the posterior of every arc. The probability computation of the spoken terms is still the same as the N-gram lattice as shown in Eq (4).

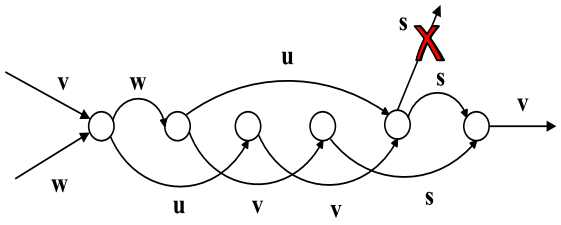
As shown in Fig. 3, there is a comparison of the pruned path processing method between the trigram lattice and unconstrained word graph. Panel (a) shows that the dead path in the rectangular filled with lined spots will discard the last N-1 arcs before the dead end node v according to the N-gram limitation. On the contrary, Panel (b) shows the unconstrained word graph structure will discard only the last hypothesis. After the UWGE process, the lattice is turned into a sausage-like form and keeps every decoding hypothesis.

The implementation of our approach is shown in Algorithm 1, and there are some properties with the unconstrained word graph:

- There is only one node in every speech frame, and the node v indicates only the time of the hypotheses. The nodes are used to record the time boundaries of the arcs. The hypotheses can be preserved in the unconstrained word graph structure since that they will not be pruned due to their dead successor hypothesis.
- The unconstrained word graph generation algorithm will not affect the time of index stage, for the reason that it is only to reconstruct the graph structure.
- There is no need to change the search strategy since the unconstrained word graph is still the structure



(a) Trigram Lat. Prune Strategy



(b) UWGE Lat. Prune Strategy

Figure 3. Comparison of Prune Strategy

Algorithm 1 unconstrained word graph generation algorithm

1: Create a node on current time frame t
2: On each frame, we consider all of the N+1 tuple $(u, v, .., s; t)$ and keep the start time boundary $\tau(u, v, ..., s; t)$
3: Create an arc $e = \{S(e), E(e), w(e), l(e)\}$ on the word graph for each N+1 tuple.
- $S(e)$ and $E(e)$ indicate the start time and end time of the hypothesis
- $w(e)$ is the weight of the arc as definition of $eq(1)$
- $l(e)$ is ID of the hypothesis

4: **if** $\exists e'$,$S(e') = n(\tau(u, v, ..., s; t))$,$E(e') = n(t)$,$l(e') = s$ **then**
5: merge the e and e'
6: **if** $w(e) < w(e')$ **then**
7: $e = e'$
8: **else**
9: retain e
10: **end if**
11: connect the arc e with start node and end node
- the start node:$S(e) = v(\tau(u, v, ..., s; t))$
- the end node:$E(e) = v(t)$

12: **end if**
13: erase the word graph and delete the dead paths [7]

suitable for other algorithms and the speed of index stage will not be affected.

- The arcs with the same start and end nodes will be merged together and this may lead to a accurate loss [12].

VI. DYNAMIC POSITION SPECIFIC POSTERIOR LATTICE

In this section, we will introduce the base position specific posterior lattice and our D-PSPL algorithm.

A. Position-Specific Posterior Lattices (PSPL)

The basic idea of PSPL is to calculate the posterior probability prob of a word W at a specific position pos in a lattice for a spoken segment d as a tuple $(W, d, pos, prob)$. Such information is actually hidden in the lattice L of d since in each path of L we clearly know each words position. Since it is very likely that more than one path includes the same word in the same position, we need to aggregate over all possible paths in a lattice that include a given word at a given position.

A variation of the standard forward-backward algorithm can be employed for this computation. The forward probability mass $\alpha(W, t)$ accumulated up to a given time t at the last word W needs to be split according to the length l measured in the number of words:

$$\alpha(W, t, l) = \sum_{\pi} P(\pi) \tag{5}$$

where π is a partial path in the lattice, the t means the hypothesis ends at time t and l represent the hypothesis contains l words. The backward probability $\beta(W, t)$ retains the original definition.

The position specific posterior probability for the word W being the l^{th} word in the lattice is then:

$$P(W, l|L) = \sum_{t} \frac{\alpha(W, t, l)\beta(W, t)}{\beta_{start}} Adj(W, t) \tag{6}$$

where β_{start} is the sum of all path scores in the lattice, and $Adj(W, t)$ consists of some necessary terms for probability adjustment, such as the removal of the duplicated acoustic model scores on W and the addition of missing language model scores around W. In this paper, we regard the tuples $(W, d, pos, prob)$ for a specific spoken segment d and position pos as a cluster, which in turn includes several words along with their posterior probabilities.

B. Basic Construction Units

The construction of PSPL is based on paths in a lattice. We first enumerate all the paths in the lattice, each with its own length (counted in words) and path weights as combined language and acoustic model scores. The posterior probability of a given word at a given position is then computed by aggregating all the path weights, where the paths include the given word at the given position, as the numerator and then divided by the sum of all the path weights in the lattice. The algorithm of PSPL is an

efficient way to accomplish this. We thus regard the words in each position. It is clear that the reason for the words being in the k_{th} cluster is that there exist some paths carrying those words as the k_{th} word in the paths.

C. Posterior Probabilities and Number of Clusters

In PSPL we assign a posterior probability $prob$ to a word W in the k_{th} cluster as the ratio of the sum of weights of those paths carrying W as the k_{th} word to the sum of all path weights in the lattice.

If we have K clusters in the PSPL structure, all we can say is that the longest paths (counted in words) in the lattice have K words, thus usually K is much larger than the real number of words.

D. The Motivation of Dynamic Position Specific Posterior Lattice Algorithm

In Sec. 5, we have discussed the UWGE algorithm which keep all the paths in the unconditioned word graph. But the confidence measurement in the graph is not accurate and this will lead to the degradation in the high score area. So we propose another algorithm based on the PSPL structure. We observed that the decoding procedure has the some same property as the PSPL. The PSPL algorithm considers only the position of the hypothesis in the path and do not need the time boundary. In the decoding procedure, when a hypothesis is activated, we can know the its position in the path, and we could record this information to build a dynamic PSPL. Each hypothesis once be activated, it could be kept in the dynamic structure.

In the dynamic PSPL structure, each unit is a tuple $W, t, l, score$. The W is word ID, t is the time of the hypothesis, the l means the position of the hypothesis, and the $score$ is the interpolation of AM score and LM score.

E. System Architecture of D-PSPL

Different from the base syllable system, we use both the trigram lattice and the D-PSPL. The D-PSPL structure contains all the hypotheses activated in the decoding procedure, as a result, the recall rate of the system will be improved a lot. As the UWGE algorithm, the confidence measurement is not accurate as well. So we make use of the traditional trigram lattice. We convert the lattice into PSPL and combine it with the D-PSPL. The hypotheses of the two structure will be merged together. If the hypothesis in both structure, we interpolate the PSPL posterior with the D-PSPL score. On the contrary, if the hypothesis only exists in the D-PSPL structure, the score is used as the final score.

As shown in Fig. 4, the indexing stage generates both trigram lattice and dynamic position specific posterior lattice. When the decoding procedure is over, the trigram is convert into PSPL structure. Then, the indexer combines the PSPL with D-PSPL and compress them into inverted index to restore.

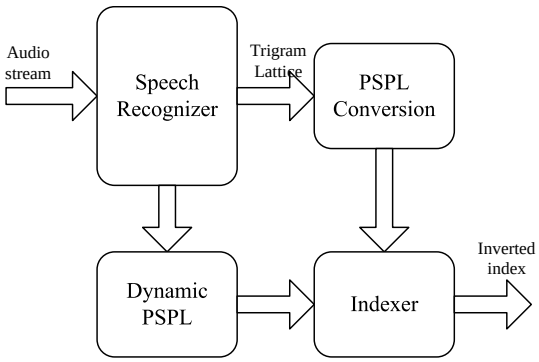


Figure 4. Frame Work of D-PSPL System

At the searching stage, the searching strategy is exactly the same to retrieve the spoken terms.

The system architecture is shown in Fig. 4.

F. Dynamic Position Specific Posterior Lattice Algorithm

Consider a word W corresponding to an edge e starting at time t_s and ending at time t_e in a searching space. It is activated during the decoding procedure. During ASR we may record the boundaries. Then, we add this word hypotheses to the D-PSPL structure. There are many clusters in the structure according to the longest path length. We record the position of the word hypothesis in its' own path and add it to the corresponding cluster. Since there is merge procedure in the decoding strategy, if the hypothesis appears in two paths, we may record its' position separately in different paths add it to different clusters. If there is same hypothesis in the different path and has the same position. If the hypotheses have different time range, we keep the only hypotheses in the cluster, otherwise, we keep the two hypotheses in the same cluster.

The confidence measurement of the hypotheses in the D-PSPL could not be calculated with forward-backward algorithm. And we use frame-synchronous measures and local measures to evaluate the confidence of the hypotheses' confidence measurement.

The frame-synchronous confidence measures use only the data available up to the frame currently being processed by the recognition engine. Thus, as soon as a frame of signal is processed by the engine, a confidence value can be computed for all the words ending at this frame. Let $[w, \tau, t]$ be the word w which starts at frame τ and ends at frame t.

The confidence measures are based on the likelihood ratio between the word w for which we want to evaluate the confidence (named current word in the following) and a set E of competing words belonging to the word graph. The following equation defines this generic likelihood ratio:

$$C([w, \tau, t]) = \frac{P(O|w)P(w)}{\sum_{w' \in E} p(O|w')P(w')} \qquad (7)$$

$P(O|w)$ is the acoustic probability of the observation sequence O given the word w and P(w) is the linguistic probability of w.

Because the frame-synchronous measures algorithm use only information of a few frames. It is not very accurate. So we use another on-the-fly confidence measurement called local measurement:

The local measures are based on the estimation of the posterior probability of words. The local confidence measures can use data slightly posterior to the word being analyzed. However this data is limited to the local neighborhood of this word and the confidence estimation does not need the recognition of the whole sentence. Thus, a short delay is introduced to allow the data to be available for computing the measures. The idea of the local measure is to define a neighborhood around the analyzed word $[w, \tau, t]$ by taking into account a fixed number of frames before and after the word. Thus, the total size of the neighborhood V of a word w is defined by the sum of the length of w and the length of both past and future neighborhood. Figure 1 shows such a neighborhood V of w with a past neighborhood of length x and a future neighborhood of length y. The durations of the past and future neighborhood measure of w. are independent. This allows us to use more data from the past (already processed, so available) without increasing the delay introduced by the future neighborhood. From the word graph generated by the recognition engine, we extracted the sub-graph corresponding to V and computed the estimation of the posterior probability of w. This estimation is obtained by the word level forward-backward method summarized by the following equations, for a bigram language model.

In this paper, we use the history of the hypothesis to calculate the local measurement.

The implement of D-PSPL algorithm is shown in Algorithm 2.

Algorithm 2 Dynamic Position Specific Posterior Lattice Generation Algorithm

Create a empty PSPL structure
Create a node on current time frame t
On each frame, we consider all of the N+1 tuple $(u, v, .., s; t)$ and keep the start time boundary $\tau(u, v, ..., s; t)$
Add the tuple to the PSPL structure according its' position information .
- $S(e)$ and $E(e)$ indicate the start time and end time of the hypothesis
- $w(e)$ is the weight of the arc as definition of $eq(1)$
- $l(e)$ is ID of the hypothesis

if $\exists e'$,$S(e') = n(\tau(u, v, ..., s; t))$,$E(e') = n(t)$,$l(e') = s$ and the position information is same **then**
  merge the e and $e^{'}$
  if e and $e^{'}$ have different time range **then**
    keep e to the cluster
  else
    retain e
  end if
  Calculate the confidence measurement of e.
end if
erase the word graph and delete the dead paths [7]

VII. Performance Evaluation

A. Evaluation setup

We conduct the experiments on two mandarin conversation telephone speech(CTS) sets. $Set1$ is 1 hour long, and a hundred keywords are selected from this corpus with 397 occurrences. This set is recorded in in laboratory environment and the speakers do not have strong accent. $Set2$ is 10 hours long CTS set with a hundred keywords and 1934 occurrences. In this set, the speakers have strong accent and the record environment has background noises.

For speech recognition, we use the one-pass recognizer [14]. The acoustic model is trained with 400 hours of CTS data with 39 dimension features (13-dimension PLP and their first and second order time derivatives). Meanwhile, a dictionary of 1276 syllables is used for our decoding. Besides, a syllable trigram language model is trained with the transcript of the acoustic model training data. We tuned the search beam width of the recognizer to 90 and the speed of the system equals 0.36xRT.

The system performance of spoken term detection is measured in 3 metrics:

a) Equal error rate (EER): It is defined as a point in DET curve where false alarm rate (FA) equals to false reject rate (FR).

b) Max recall rate (MaxRecall): It is the recall rate of all the keywords which are found by the system.

B. Experimental results

As shown in Table 1, we compare the performance of different algorithm: The trigram lattice, position specific posterior lattice, unconditioned word graph, dynamic position specific posterior lattice with frame-synchronous confidence measures and dynamic position specific posterior lattice with local measurements.

TABLE I.
The keyword spotting performance comparison.

System	EER	MaxRecall
$Set1$		
Trigram Lat.	43.47	56.51
PSPL	43.84	57.01
UWGE Lat.	38.79	63.82
D-PSPL(frame-synchronous confidence)	39.33	62.75
D-PSPL(local confidence)	38.21	62.75
$Set2$		
Trigram Lat.	52.89	48.97
PSPL	52.35	49.21
UWGE Lat.	48.35	55.76
D-PSPL(frame-synchronous confidence)	48.02	55.89
D-PSPL(local confidence)	47.16	55.89

In Table 1, we can see the trigram lattice and PSPL has the similar performance, and the UWGE algorithm

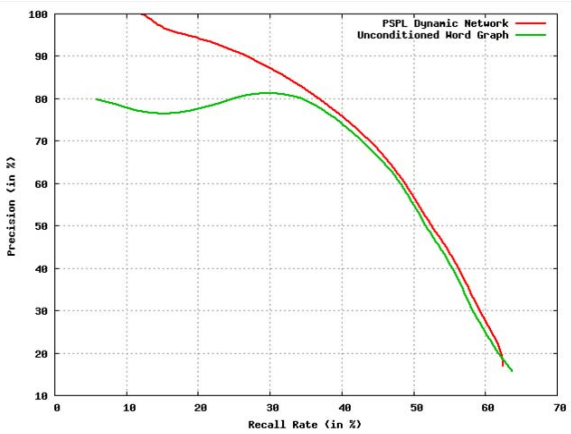


Figure 5. The Comparison of UWGE and D-PSPL

has much better EER score and max recall rate. The D-PSPL algorithm has the same performance with the UWGE algorithm.

With different confidence measurements, the local confidence is a little better than the frame-synchronous confidence. In this work, we choose to calculate the local confidence from the sentence beginning to the current frame, and we observe the local measurement is near the confidence measurement which is calculated with forward-backward algorithm.

The UWGE and D-PSPL algorithm have similar performance on EER score and max recall rate, but they are different in the high score area. As shown in Fig. 5 the comparison on the accuracy-recall-rate curve, we can observe the D-PSPL has the parts of the offline PSPL hypotheses and the the accuracy is 100% when the recall below 13%. On the contrary, the performance of UWGE algorithm has only the highest accuracy of 80%.

VIII. Conclusions

In this paper, we focused on the performance degradation of the fast keyword spotting system and addressed the problem of how to get more information generated in the decoding procedure. We aimed to get enough information from the decoding procedure directly and proposed two methods: the unconditioned word graph algorithm and dynamic position specific posterior lattice algorithm. The unconstrained word graph expansion (UWGE) to eliminate the limitation of N-gram lattice and generate unconstrained word graph, which is still fine graph structure suitable for the forward-backward algorithm and improves the recall rate of keywords.

The dynamic position specific posterior lattice(D-PSPL) algorithm is based on PSPL structure, and we use a dynamic generation algorithm to produce linear index structure. Then, we combine the D-PSPL structure and the offline PSPL structure. The confidence measurement is calculated by two on-the-fly confidence measurements: frame-synchronous confidence measures and local measurements.

On our test sets, we compared the two algorithms with trigram lattice and PSPL. The proposed algorithms outperform and the D-PSPL method has better accuracy when the confidence score threshold comes higher.

In future works, we will investigate the problem of the inaccurate language model information kept in the unconstrained word graph and improve the D-PSPL confidence measurement.

Acknowledgment

This work is partially supported by the National Natural Science Foundation of China (Nos. 10925419, 90920302, 61072124, 11074275, 11161140319, 91120001, 61271426), the Strategic Priority Research Program of the Chinese Academy of Sciences (Grant Nos. XDA06030100, XDA06030500), the National 863 Program (No. 2012AA012503) and the CAS Priority Deployment Project (No. KGZD-EW-103-2).

References

[1] I. Szoke, P. Schwarz, P. Matejka, L. Burget, M. Karafiát, M. Fapso, and J. Cernocky, "Comparison of keyword spotting approaches for informal continuous speech," in *Ninth European Conference on Speech Communication and Technology*, 2005.

[2] T. Mertens and D. Schneider, "Efficient subword lattice retrieval for german spoken term detection," in *Acoustics, Speech and Signal Processing, 2009. ICASSP 2009. IEEE International Conference on*. IEEE, 2009, pp. 4885–4888.

[3] P. Yu, K. Chen, C. Ma, and F. Seide, "Vocabulary-independent indexing of spontaneous speech," *Speech and Audio Processing, IEEE Transactions on*, vol. 13, no. 5, pp. 635–643, 2005.

[4] F. Seide, P. Yu, and Y. Shi, "Towards spoken-document retrieval for the enterprise: Approximate word-lattice indexing with text indexers," in *Automatic Speech Recognition & Understanding, 2007. ASRU. IEEE Workshop on*. IEEE, 2007, pp. 629–634.

[5] S. Ortmanns, H. Ney, and X. Aubert, "A word graph algorithm for large vocabulary continuous speech recognition," *Computer Speech & Language*, vol. 11, no. 1, pp. 43–72, 1997.

[6] S. Ortmanns and H. Ney, "The time-conditioned approach in dynamic programming search for lvcsr," *Speech and Audio Processing, IEEE Transactions on*, vol. 8, no. 6, pp. 676–687, 2000.

[7] H. Ney and S. Ortmanns, "Progress in dynamic programming search for lvcsr," *Proceedings of the IEEE*, vol. 88, no. 8, pp. 1224–1240, 2000.

[8] M. Saraclar and R. Sproat, "Lattice-based search for spoken utterance retrieval," *Urbana*, vol. 51, p. 61801, 2004.

[9] G. Evermann and P. Woodland, "Posterior probability decoding, confidence estimation and system combination," in *Proc. Speech Transcription Workshop*, vol. 27. Baltimore, 2000.

[10] F. Wessel, R. Schluter, and H. Ney, "Using posterior word probabilities for improved speech recognition," in *Acoustics, Speech, and Signal Processing, 2000. ICASSP'00. Proceedings. 2000 IEEE International Conference on*, vol. 3. IEEE, 2000, pp. 1587–1590.

[11] H. Jiang, "Confidence measures for speech recognition: A survey," *Speech communication*, vol. 45, no. 4, pp. 455–470, 2005.

[12] T. Kemp and T. Schaaf, "Estimating confidence using word lattices," in *Fifth European Conference on Speech Communication and Technology*, 1997.

[13] F. Wessel, R. Schluter, K. Macherey, and H. Ney, "Confidence measures for large vocabulary continuous speech recognition," *Speech and Audio Processing, IEEE Transactions on*, vol. 9, no. 3, pp. 288–298, 2001.
[14] S. Jian, L. Ta, Q. Zhang, Z. Qingwei, and Y. Yonghong, "A one-pass real-time decoder using memory-efficient state network," *IEICE TRANSACTIONS on Information and Systems*, vol. 91, no. 3, pp. 529–537, 2008.

Zhen Zhang received his B.E. degree in Electronic Information Engineering from Tianjin University in 2007. Now he is a doctor candidate of the Key Laboratory of Speech Acoustics and Content Understanding, Institute of Acoustics (IOA), Chinese Academy of Sciences (CAS). His research interests include Large Vocabulary Continuous Speech Recognition, Spoken Term Detection, Decoding Search Algorithms, and Machine Learning.

Ji Xu received his Bachelor and Master's degrees in 2008 and 2011 respectively, from the Department of Electronic Engineering, Tsinghua University. Now he is a doctor candidate of the Key Laboratory of Speech Acoustics and Content Understanding, Institute of Acoustics (IOA), Chinese Academy of Sciences (CAS). His research interest covers speech recognition, speech synthesis and artificial intelligence.

Yujing Si received his B.E. degree in Information Engineering from Jilin University in 2009. Now he is a doctor candidate of the Key Laboratory of Speech Acoustics and Content Understanding, Institute of Acoustics (IOA), Chinese Academy of Sciences (CAS). His research interests include Large Vocabulary Continuous Speech Recognition, Language Model, decoding search algorithms, and automatic pronunciation evaluation.

Qingwei Zhao received his Ph.D degree in Electronic Engineering from Tsinghua University in 1999. Now he is Professor of the Key Laboratory of Speech Acoustics and Content Understanding. His research interests include speech processing and recognition, keyword spotting, and human computer interface.

Yonghong Yan received his B.E. from Tsinghua University in 1990, and his PhD from Oregon Graduate Institute (OGI). He worked in OGI as Assistant Professor (1995), Associate Professor (1998) and Associate Director (1997) of Center for Spoken Language Understanding. He worked in Intel from 1998-2001, chaired Human Computer Interface Research Council, worked as Principal Engineer of Microprocessor Research Lab and Director of Intel China Research Center. Currently he is a professor and director of the Key Laboratory of Speech Acoustics and Content Understanding. His research interests include speech processing and recognition, language/speaker recognition, and human computer interface. He has published more than 100 papers and holds 40 patents.