Unconstrained Word Graph Based Keyword Spotting

Zhen Zhang, Yujing Si, Yong Liu, Qingwei Zhao, Yonghong Yan
The Key Laboratory of Speech Acoustics and Content Understanding
Chinese Academy of Sciences, Beijing 100190, P.R.China
{zhangzhen , siyujing , liuyong, 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 an approach named unconstraint word graph expansion (UWGE) to keep the pruned hypotheses which are discarded in the decoding procedure but may contain correct hypotheses. The proposed approach 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 UWGE with that on traditional trigram lattice , and our approach gives satisfying performance gains over trigram lattice. We also show the relationship between the performance and the system speed based on this approach.**

Keywords- spoken term detection, unconstraint word graph expansion, N-gram lattice limitation

I. INTRODUCTION

The ever growing volume of recorded speech data collect-ed from telephones, cell phones and internet conversation etc, poses great challenge for the spoken language processing te-chnologies . Keyword spotting is a very important branch of speech recognition, which is the task of detecting the occur-rences of predefined keywords in the unconstrained audio s-tream.

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 an approach named unconstrained word graph expansion (UWGE). This is done by rebuilding the N-gram lattice into another form: unconstrained word graph. 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 shows the architecture of our system; In Section 3, we discuss the lattice generation algorithm and baseline keyword spotting paradigm; Section 4 describes the limitation of the N-gram lattice and proposed method in detail; Experimental results are presented in Section 5, followed by conclusions in Section 6.

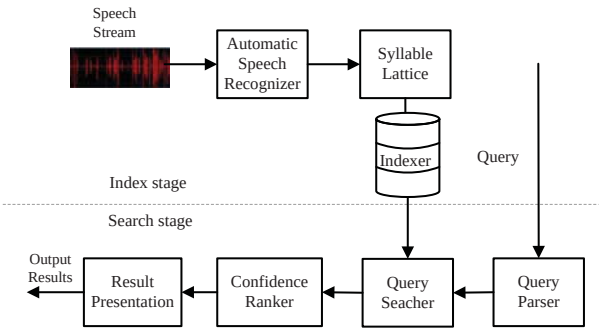


Figure 1. The framework of the system

II. SYSTEM ARCHITECTURE

Figure 1 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.

III. 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.

Published by Atlantis Press, Paris, France.

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 weighted directed acyclic graph (DAG) [5]. It is defined as $G = \{V, E, W, L, v_{start}, v_{end}\}$ where arcs E represent the syllable hypotheses with recognizer weight W and ID L, and nodes V are the connections between them, encoding times and N-gram language model state. v_{start} and $v_{end} \in V$ 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 an 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 unit, $\{s_j, j = 1, 2, ..., Q\}$. 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:

$$q_{v_s,s,v_e} = p^{\frac{1}{\lambda}}(O(t_{v_s} ... t_{v_e}) \mid v_s, s, v_e) \bullet p(s \mid v_s) \qquad (1)$$

Where $p(O(t_{v_s} ... t_{v_e}) \mid v_s, s, v_e)$ 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). $p(s \mid v_s)$ 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 $*\text{-}t_s\text{-}s\text{-}t_e\text{-}*$ denotes the set of paths which contain s with boundaries ts and te. To compute it, we sum all nodes (v_s, v_e) with given time points (t_s, t_e):

$$p(*\text{-}t_s\text{-}s\text{-}t_e\text{-}* \mid O) = \sum_{\substack{(v_s,v_e): \\ t_{v_s}=t_s \wedge t_{v_e}=t_e}} p(*\text{-}v_s\text{-}s\text{-}v_e\text{-}* \mid O) \qquad (2)$$

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

$$p(*\text{-}v_s\text{-}s\text{-}v_e\text{-}* \mid O) = \frac{\alpha_{v_s} \cdot q_{v_s,s,v_e} \cdot \beta_{v_e}}{\beta_{enter}} \qquad (3)$$

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]. v_{enter} is the total probability over all paths . So, the probability of the spoken terms can be computed as:

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

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

IV. PROPOSED METHOD

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

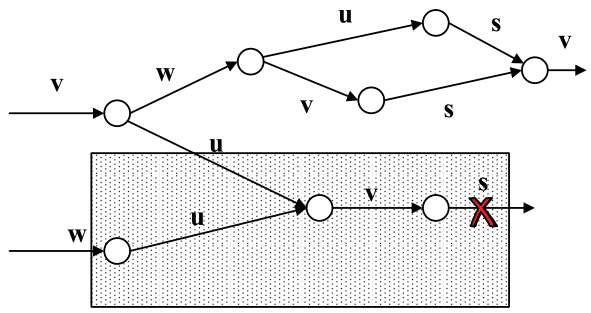
A. *The limitation of N-gram lattice*

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. However, on the other hand, it makes the lattice unable to preserve the dead paths which mean the paths can't 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.

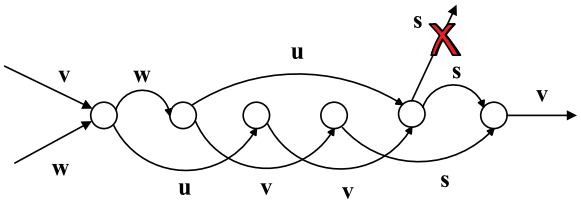
Meanwhile, due to the limitation of the N-gram model, each hypothesis is an N+1 tuple $(u, v, ..., s; t)$. When the pruning happens, this N+1 tuple will be pruned from the lattice. However, the influence of path prune is a bit too long, and it affects not only the last hypothesis but also a series of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

Published by Atlantis Press, Paris, France.



(a) Trigram Lat. prune strategy



(b) UWGE Lat. prune strategy

Figure 2. Comparison of prune strategy

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 $\forall v \in V$ 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 Figure 2, 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.

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
- $w(e)$ 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]

The implementation of our approach is shown in Algorithm1 and there are some properties with the unconstrained word graph:

- 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 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 .

V. EXPERIMENTS

A. *Evaluation setup*

We conduct the experiments on two mandarin conversation telephone (CTS) speech sets. *Set1* is 1 hour long, and a hundred keywords are selected from this corpus with 397 occurrences. This set is recorded 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 Onepass recogni-zer. 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). 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) Figure of merit (FOM): The NIST Figure Of Merit defined as the dectection/false-alarm curve averaged over [0..10] false alarms per keyword per *h* hours.

c) 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, based on the proposed method, the UWGE based keyword spotting performance gets 10.61% reduction on EER score and 9.29% improvement on FOM score relatively on average compared to the trigram lattice, and the MaxRecall has a 6.76% relative improvement than trigram lattice. The more information in the unconstrained word graph is the main reason for these improvements.

TABLE I. THE KEYWORD SPOTTING PERFORMANCE COMPARISON.

System	EER	FOM	MaxRecall
Set1			
Trigram Lat.	45.5	54.5	54.5
UWGE Lat.	39.7	59.52	60.3
Set2			
Trigram Lat.	59.77	39.25	40.23
UWGE Lat.	54.71	42.99	47.9

We also compare the performance of the graph structure between unconstrained word graph and trigram lattice on *Set1*. We use the following metrics:

a) The average number of hypotheses on each frame (Avr.#WE) : It is the average number of hypotheses generated during the decoding procedure on each frame.

b) The average number of hypotheses kept in the graph on each frame (Avr.#Trace) : It is the average number of hypotheses kept in the graph on each frame. It may not equal to the number of arcs on each frame due to the merge procedure of the UWGE algorithm.

c) Graph word error rate (GER): It is computed by determining that sentence through the word graph that best matches the spoken sentence.

TABLE II. THE LATTICE PERFORMANCE COMPARISON.

system	Avr.#WE	Avr.#Trace	GER
Trigram Lat.	9.81	0.10	29.48
UWGE Lat.	9.81	0.72	22.69

In Table 2, we can see the unconstrained word graph has the same Avr.#WE with trigram lattice due to the same decoding procedure but higher Avr.#Trace which means more hypotheses are kept in the unconstrained word graph. The GER of unconstrained word graph is much lower than that of trigram lattice and this indicates that unconstrained word graph contains more useful information.

VI. CONCLUSIONS

In this work, 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 an approach named 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. On our test sets, we compared unconstrained word graph with trigram lattice. The unconstrained word graph achieved better performance than trigram lattice on graph error rate (GER) etc. In the spoken term detection task, the experiments showed the unconstrained word graph gets better performance than trigram lattice in the fast keyword spotting system.

ACKNOWLEDGMENT

This work is partially supported by the National Natural Science Foundation of China (Nos. 10925419, 90920302, 61072124, 11074275, 11161140319, 91120001) and the Strategic Priority Research Program of the Chinese Academy of Sciences (Grant Nos. XDA06030100, XDA06030500).

REFERENCES

[1] I. Szoke, P. Schwarz, P. Matejka, L. Burget, M. Karafi´at, 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.