QUERY-BASED COMPOSITION FOR LARGE-SCALE LANGUAGE MODEL IN LVCSR

Yang Han, Chenwei Zhang, Xiangang Li, Yi Liu, Xihong Wu

Speech and Hearing Research Center
Key Laboratory of Machine Perception(Ministry of Education)
Peking University, Beijing, 100871, China
{hanyang, zhangcw, lixg, liuy, wxh}@cis.pku.edu.cn

ABSTRACT

This paper describes a query-based composition algorithm that can integrate an ARPA format language model in the unified WFST framework, which avoids the memory and time cost of converting the language models to WFST and optimizing the WFST of language models. The proposed algorithm is applied to on-the-fly one-pass decoder and rescoring decoder. Both modified decoder require less memory during decoding on different scale of language models. What's more, query-based on-the-fly one-pass decoder nearly has the same decoding speed as standard one and query-based rescoring decoder even use less time to rescore the lattice. Because of these advantages, large-scale language models can be applied by query-based composition algorithm to improve the performance of large vocabulary continuous speech recognition.

Index Terms— large-scale language model, query-based, composition, WFST, speech recognition

1. INTRODUCTION

Weighted finite-state transducer (WFST) approach[1] has become a promising alternative formulation to lexical prefix tree approach in speech recognition. It offers a unified framework representing various knowledge sources and produces a full search network optimized up to the HMM states. The WFST approach not only provides a flexible way of integrating multiple knowledge sources but also offers efficient optimization algorithms that make very good use of the sparsity of the integrated network[2]. The reason is that operations such as determination and minimization achieve global optimization over the whole search space while lexical prefix tree approach is limited to a local optimization corresponding to each models[3]. Moreover, such static network decoders are known to be very efficient in terms of computation time.

However, the disadvantage of WFST is related to the cost of memory. The WFST network resulting from fully composition of all knowledge sources can be very large especially involved in large-scale language models, thus becoming an unmanageable graphs. Moreover, construction and optimization algorithms of WFST require a large quantity of memory which prevents researchers from applying large-scale language models. A straight strategy to reduce the oversized network is to employ multiple (usually two) recognition passes based on lattice[4]. Another strategy relying on a single pass called on-the-fly composition[5] is a practical alternative to a fully composed single WFST.

It has been widely observed that the effectiveness of statistical natural language processing techniques is highly susceptible to the data size used to develop them[6]. Towards this end, there have been considerable efforts in gathering ever larger datasets, culminating the release of Gigaword corpus, 1 Tera-word Google N-gram and Microsoft Web N-gram corpus[6]. Considering the benefits of large datasets and the convenience to get them, researchers would like to make use of large datasets to train large scale models and apply them in the systems. Moreover, for language models, some tools such as KenLM[7], SRILM[8], IRSTLM[9] all bring the convenience to access large-scale language models.

As language models are getting larger and larger, methods mentioned above to relieve the problem of memory expansion in WFST approach are not as effective as before. That is because the large-scale language models are much larger than before which makes the remainder language models are still too large to handle.

In this paper, we propose a query-based composition algorithm that can integrate an ARPA format language model in the unified WFST framework which avoids the memory and time cost of converting the language models to WFST and optimizing the WFST of language models. What's more, the memory cost during query-based composition is less than the standard composition algorithm. These advantages make it easier to apply large-scale language models in large vocabulary continuous speech recognition based on WFST approach. The query-based composition algorithm is applied to on-the-fly one-pass decoder and rescoring decoder based on lattice.

This paper is organized as follows. In Section 2 prior work which relieves the memory expansion problem of WFST is reviewed. Section 3 introduces query-based composition al-

978-1-4799-2893-4/14/$31.00 ©2014 IEEE

gorithm. Section 4 describes the experiments and results. The final section summarizes the main conclusions.

2. RESCORING STRATEGY AND ON-THE-FLY ONE-PASS STRATEGY

When researchers employ the WFST approach in speech recognition, memory cost is the obstacle that prevents researchers from using large-scale language models. It has been observed that the size of the resultant transducer is roughly proportional to the size of the language model. So the explicit integration of the lexicon with large-scale language models generates very large unmanageable graphs, and has been regarded as applicable only to small bigram language models[10]. This leads to the focus on the composition of lexicon and language models[11][12][13].

The straight idea is multiple-pass (usually two-pass) strategy, using a simplified language model to constrain the network and applying the most detailed language model in a second rescoring pass. This strategy firstly constrains the search space to a lattice and secondly searches in the lattice in the second pass using the most accurate language models. Multiple-pass strategy reduces the size of search space pass by pass, rather than directly searching the results in a very large search space. So it can reduce the memory cost during decoding.

Another strategy employing a single pass called on-the-fly composition. The full language model is factorized into a small language model and a difference language model. The small language model is incorporated with the other knowledge sources offline. The full language model information is integrated in runtime by composing the static optimized network with the difference language model on-the-fly. The small language model scores incorporated with the other knowledge sources can be interpreted as language model lookahead[5]. This strategy can also save memory during decoding since only the small language model is incorporated with the other knowledge sources statically and the difference language model is integrated on-the-fly. In this paper, both the most detailed language model used in final pass and the difference language model used by on-the-fly decoder are called remainder language model.

With the popularity of the Internet, more and more information is created on the Internet. It is much more convenience to gather and access large amount of data. As known to all, for statistical language models, large amount of training data can cover more linguistic phenomena and thus improve the performance. There is more training data and thus models are larger than before. However, when these large-scale language models are used, the memory cost cannot be tolerated even using on-the-fly composition or rescoring strategies. Moreover, no matter rescoring or on-the-fly composition strategy, they both need to convert the language models into WFSTs which consumes much time and memory. This paper proposes the query-based composition algorithm that can avoid these consumption.

Algorithm 1 query-based composition

input:
T_1: input WFST
G_s: small language model
G_r: remainder language model
output:
T_1: output WFST whose scores are updated by G_r

$I_2 \leftarrow$INIT(G_s, G_r);
$Q \leftarrow I \leftarrow S \leftarrow I_1 \times I_2$;
while $S \neq \varnothing$ **do**
  $(q_1, q_2) \leftarrow$HEAD(S);
  DEQUEUE(S);
  if $q_1 \in F_1$ **then**
    INSERT$(F, (q_1, q_2))$;
    $\rho(q_1, q_2) \leftarrow \rho_1(q_1)$;
  end if
  for all $e_1 \in E[q_1]$ **do**
    $h \leftarrow$FINDHISTORY(q_2);
    $(nh, c) \leftarrow$FULLSCORE$(o[e_1], h, G_s, G_r)$;
    $q_2' \leftarrow$CREATESTATE(nh);
    if $(n[e_1], q_2') \notin Q$ **then**
      INSERT$(Q, (n[e_1], q_2'))$;
      ENQUEUE$(S, (n[e_1], q_2'))$;
      $\omega' \leftarrow \omega[e_1] \otimes c$;
      INSERT$(E, ((q_1, q_2), i[e_1], o[e_1], \omega', (n[e_1], q_2')))$;
    end if
  end for
end while
return T_1;

3. THE QUERY-BASED COMPOSITION ALGORITHM

By analyzing on-the-fly composition and rescoring strategy, we can get that in both strategies the role of the remainder language models is just to update the language model scores which have been smeared along the lattice or the static network by simplified or small language models. There are many ways to query the language model scores in the remainder language models, not only in the form of WFST. We don't have to convert the remainder language models into WFSTs, wasting the time and memory to optimize and store them. Furthermore if the remainder language models are very large, the time and memory cost of composing it with the other WFSTs cannot be tolerated which also limits the usage of large-scale language models in the approach of WFST. The query-based composition algorithm can integrate the lattice and the static network with ARPA format language model directly. It differs from standard composition algorithm by querying the language model scores based on partial word hypothesis and

output labels during composition. Here we take the rescoring strategy as an example to illustrate the problem. During the composition of a lattice with the remainder language model in the form of WFST, the weight of partial word hypothesis g in the lattice is updated by the transition e in the WFST of remainder language models as:

$$\alpha[g'] = \alpha[g] + \omega[e] \tag{1}$$

where $\alpha[g]$ is the score of partial word hypothesis, $\omega[e]$ is the weight of transition e in remainder language models, and $\alpha[g']$ is the new score of partial word hypothesis in the lattice after update.

When the remainder language models are not in the form of WFST, a language model score correction, $c[o[e], h[g]]$, is applied:

$$\alpha[g'] = \alpha[g] + c[o[e], h[g]] \tag{2}$$

The value of $c[o[e], h[g]]$ is the difference between the small language language model score $log(P_{sm}(o[e]|h[g]))$ and the remainder language model score $log(P_{re}(o[e]|h[g]))$, where $h[g]$ is the output symbol sequence of the hypothesis g:

$$\begin{aligned} c[o[e], h[g]] = log(P_{sm}(o[e]|h[g])) - \\ log(P_{re}(o[e]|h[g])) \end{aligned} \tag{3}$$

Algorithm 1 is the pseudocode of the query-based composition algorithm. The input is a WFST T_1, a small language model G_s and the remainder language model G_r. I is the set of initial states and F is the set of final states. E is a finite set of transitions and $E[q]$ denotes the set of transitions leaving state q. Given a transition $e \in E$, $p[e]$ denotes its origin state, $n[e]$ denotes its destination state, $i[e]$ its input label, $o[e]$ its output label and $\omega[e]$ its weight. λ denotes an initial state weight assignment and ρ denotes a final state weight assignment. The remainder language model scores are queried by partial word hypothesis h and output label $o[e_1]$. And the queried scores are used to update the lattice and the static network. The query-based composition algorithm returns the transducer T_1 whose scores have been updated by the remainder language model G_r. The language model scores are queried by the open source toolkit KenLM[7].

4. EXPERIMENTS

Firstly, the query-based composition algorithm is applied to on-the-fly one-pass decoder and the experimental results are contrasted with the standard on-the-fly one-pass decoder's. Secondly, we apply the query-based composition algorithm to rescoring decoder and compare the results with the standard rescoring decoder's. The baseline are both the open source project KALDI[14].

4.1. Experimental Setup

We evaluate the query-based composition algorithm with corpus made up of broadcast news and broadcast conversion mandarin speech coming from 49 television broadcasts, totally around 3000-hour speech. A 3-hour speech data set is built to conduct the ASR evaluation, which are random selected from the corpus. A lexicon containing 60k words is used in all experiments. To evaluate the performance of the proposed algorithm on different scale of language models, we randomly extract 0.5GB, 1GB, 2GB and the entire data from Gigaword corpus to train different scale of trigram language models using the Modified Kneser-Ney smooth. The detailed information of these language models are presented in Table 1. Since all the scores obtained by different composition algorithms are always the same, the character error rate(CER) of different composition algorithms are always the same. That's why we only evaluate the proposed algorithm in the aspect of time and memory cost.

Table 1. different scale of language models

language model	n-grams	file size(GB)
Giga_0.5GB	40,192,463	0.95
Giga_1GB	68,369,394	1.60
Giga_2GB	149,342,585	3.49
Giga_all	1,175,519,733	26.20

4.2. Experimental Results

When compared with standard on-the-fly one-pass decoder, the query-based one-pass decoder outperforms in the aspect of memory consumption. Table 2 shows the performance of these two one-pass decoders on different scale of language models. The decoding time of one-pass decoder is represented by RTF that indicates the ratio of decoding time to utterance time. Query-based one-pass decoder uses about 61%-68% memory of standard on-the-fly one-pass decoder. Meanwhile, the decoding speed of these two decoders are nearly the same. For standard on-the-fly one-pass decoder, if using the Giga_all language model, we firstly should convert the language model to WFST and then during the composition we also need about two times memory of the size of language model[3]. Limited by the size of our system's memory, standard on-the-fly one-pass decoder cannot make use of Giga_all language model. While query-based one-pass decoder can apply Giga_all language model within the memory space limit.

When applying query-based composition algorithm to rescoring decoder, query-based rescoring decoder outperforms in both aspect of memory consumption and time cost. To better evaluate the performance of rescoring decoder, we only calculate the rescoring time of two rescoring decoders, exclude the time of generating lattice and loading models. Table 3 reveals the performance of these two rescoring decoders. As shown in the Table 3, the query-based rescoring decoder uses less memory and less time to rescore the lattice. The memory consumption of query-based rescoring decoder is 49%-57% of standard rescoring decoder and time

Table 2. performance of on-the-fly decoders

language model	decoder	RTF	memory(GB)
Giga_0.5GB	standard	0.82	2.35
	query-based	0.86	1.62
Giga_1GB	standard	0.79	3.28
	query-based	0.84	2.11
Giga_2GB	standard	0.79	5.70
	query-based	0.87	3.53
Giga_all	standard	-	-
	query-based	0.86	21.08

cost is 78%-89% of standard rescoring decoder. For the same reason, standard rescoring decoder still can not use Giga_all language model.

Table 3. performance of rescoring decoders

language model	decoder	time(s)	memory(GB)
Giga_0.5GB	standard	299.17	1.57
	query-based	258.52	0.77
Giga_1GB	standard	304.42	2.32
	query-based	271.15	1.24
Giga_2GB	standard	366.52	4.38
	query-based	286.79	2.51
Giga_all	standard	-	-
	query-based	328.22	18.14

4.3. Discussions

As shown in the experimental results, query-based one-pass decoder and rescoring decoder both require less memory than the standard counterparts. And the query-based rescoring decoder even need less time to rescore the lattice but query-based one-pass decoder is a little slower than standard on-the-fly one-pass decoder. This phenomenon is caused by that standard on-the-fly one-pass decoder uses cache to store the discovered relationship between language model scores and the specific states and arcs of the language model in WFST. This implementation masks the benefit brought by the query-based composition algorithm. To confirm this reason, we remove the cache in both one-pass decoders and the result are presented in Table 4. As shown in the results, query-based decoder is faster than on-the-fly one-pass decoder without cache implementation.

On the other hand, when referring large-scale language models, researchers would like to use dynamic decoder based on prefix tree. Dynamic decoder generates just the required parts of the search network which requires less memory. However, the dynamic search network is less compact than static search network, even though employing suffix sharing[15]. Table 5 shows the performance of dynamic decoder in HTK[16] and query-based one-pass decoder when applying Giga_1GB language model with the limit of CER. Query-based one-pass decoder can decode nearly 17 times faster than dynamic decoder and only requires about 1.2 times memory of dynamic decoder. When applying large-scale language models, the memory cost of dynamic decoder and query-based one-pass decoder is proportional to the size of language model but the query-based one can decode much faster.

Table 4. performance of on-the-fly decoders without cache

language model	decoder	RTF	memory(GB)
Giga_0.5GB	standard	1.22	2.36
	query-based	1.06	1.78
Giga_1GB	standard	1.19	3.29
	query-based	1.05	2.07
Giga_2GB	standard	1.06	5.71
	query-based	0.94	3.49

Table 5. dynamic decoder vs. query-based one-pass decoder

decoder	RTF	memory(GB)	CER(%)
dynamic	17.56	1.72	11.25
query-based	0.84	2.11	10.65

5. CONCLUSION

We propose the query-based composition algorithm for integrating the ARPA format language model into the WFST framework. The experimental results show that the query-based algorithm not only saves the time and memory to convert the ARPA format language model to WFST, but also requires less memory during decoding. For query-based one-pass decoder, it can decode as fast as the standard one. And for query-based rescoring decoder, it uses less time to rescore the lattice. When compared with dynamic decoder based on prefix tree, query-based one-pass decoder is much faster with little increasement of memory. The query-based composition algorithm finds a balance between static and dynamic decoder and makes it much easier to apply large-scale language model in LVCSR.

The query-based algorithm can also be applied to other similar tasks where large-scale language models are needed, such as word segmentation based on WFST.

6. ACKNOWLEDGEMENT

The work was supported in part by the National Basic Research Program of China (2013CB329304), the research special fund for public welfare industry of health(201202001) and National Natural Science Foundation of China (No. 61121002, No.91120001).

7. REFERENCES

[1] Pereira F and Riley M, "Speech recognition with weighted finite-state transducers," in *Springer Handbook of Speech Processing*, 2008, pp. 559–584.

[2] Caseiro D and Trancoso I, "A tail-sharing wfst composition algorithm for large vocabulary speech recognition," in *Acoustics, Speech, and Signal Processing*, 2003, pp. I–356–I–359 vol. 1.

[3] Hori T, Hori C, Minami Y, and et al, "Efficient wfst-based one-pass decoding with on-the-fly hypothesis rescoring in extremely large vocabulary continuous speech recognition," *Audio, Speech, and Language Processing, IEEE Transactions*, pp. 15(4): 1352–1365, 2007.

[4] Ortmanns S, Ney H, and Aubert X, "A word graph algorithm for large vocabulary continuous speech recognition," *Computer Speech and Language*, pp. 11(1): 43–72, 1997.

[5] Dolfing H J G A and Hetherington I L, "Incremental language models for speech recognition using finite-state transducers," *Automatic Speech Recognition and Understanding*, pp. 194–197, 2001.

[6] Wang K, Thrasher C, Viegas E, and et al, "An overview of microsoft web n-gram corpus and applications," in *Proceedings of the NAACL HLT 2010 Demonstration Session*, 2010, pp. 45–48.

[7] Heafield K, "Kenlm: Faster and smaller language model queries," in *Proceedings of the Sixth Workshop on Statistical Machine Translation*, 2011, pp. 187–197.

[8] Stolcke A, "Srilm-an extensible language modeling toolkit," in *INTERSPEECH*, September 2002.

[9] Federico M, Bertoldi N, and Cettolo M, "Irstlm an open source toolkit for handling large scale language models," in *INTERSPEECH*, 2008, pp. 1618–1621.

[10] Caseiro D and Trancoso I, "On integrating the lexicon with the language model," in *INTERSPEECH*, 2001, pp. 2131–2134.

[11] Caseiro D and Trancoso I, "A specialized on-the-fly algorithm for lexicon and language model composition," *Audio, Speech, and Language Processing, IEEE Transactions*, pp. 14(4): 1281–1291, 2006.

[12] Allauzen C, Riley M, and Schalkwyk J, "A generalized composition algorithm for weighted finite-state transducers," in *INTERSPEECH*, 2009, pp. 1203–1206.

[13] Caseiro D and Trancoso I, "Using dynamic wfst composition for recognizing broadcast news," in *INTERSPEECH*, 2002.

[14] Povey D, Ghoshal A, Boulianne G, and et al, "The kaldi speech recognition toolkit," in *Proc. ASRU.*, 2011, pp. 1–4.

[15] Nolden D, Rybach D, and Ney H, "Joining advantages of word-conditioned and token-passing decoding," in *Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2012, pp. 4425–4428.

[16] Young S, Evermann G, Gales M, and et al, "The htk book(for htk version 3.4)," in *Cambridge university engineering department*, 1995.