Compact WFSA Based Language Model and Its Application in Statistical Machine Translation*

Xiaoyin Fu, Wei Wei, Shixiang Lu, Dengfeng Ke, and Bo Xu

Interactive Digital Media Technology Research Center
Institute of Automation, Chinese Academy of Sciences, Beijing, China
{xiaoyin.fu,wei.wei.media,shixiang.lu,dengfeng.ke,xubo}@ia.ac.cn

Abstract. The authors explore the fast query techniques for n-gram language model (LM) in statistical machine translation (SMT), and then propose a compact WFSA (weighted finite-state automaton) based LM motivated by the contextual features in process of model queries. It is demonstrated that the query based on WFSA can effectively avoid the redundant queries and accelerate the query speed. Furthermore, it is revealed that investigating a simple caching techni que can further speed up the query. The experiment results show that this method can finally speed up the LM query by 75% in relative. With the LM order increasing, the performance benefits by WFSA will be much more significant.

Keywords: N-gram Language Model, WFSA, Fast Query, SMT.

1 Introduction

N-gram language model is one of the important components in modern statistical machine translation systems. It helps to generate the reasonable translations which are corresponding to grammar and common usage of natural language. Using higher order LM models and more training data can significantly improve the translation performances [1]. However, decoding a single sentence can trigger hundreds of thousands of queries to the LM. Therefore, the LM must be fast for the actual SMT systems. Previous proposed techniques [2,3,4] employed the LM with trade-offs among time, space and accuracy. In this paper, we try to deal with this case by a compact WFSA based LM.

The weighted finite-state automaton has been introduced successfully in many natural language processing applications, as many of them have been possible to break down, both conceptually and literally, into cascades of simpler probabilistic finite-state transition [5]. We consider the LM query process as a sequence of state transitions in WFSA, which draws a uniform framework [6] for LM without almost any redundant operations and speeds up the queries substantially. Our WFSA based LM is organized with a *trie* structure which makes the LM stored in a compact way. We also introduce a simple LM cache by hash table to further

* This work was supported by 863 program in China (No. 2011AA01A207).

M. Zhou et al. (Eds.): NLPCC 2012, CCIS 333, pp. 154–163, 2012.

speed up the queries. The results show that our method can finally improve the query speed by about 75%.

2 Related Work

The current research efforts to speed up the query typically follow the context features of query process in n-gram LM [7,8,9]. Pauls [7] presented several language model implementations that were highly compact and fast to query. They introduced a language model that took a word-and-context encoding for the suffix of the original query to accelerate the scrolling queries. They also exploited the scrolling nature of queries in the n-grams encoded *tries* with last-rest instead of the reverse direction, although they found the speed improvement from switching to a first-rest encoding was modest. This has also been exploited by Li [8], who proposed equivalent language model states to explore the back-off property of n-gram language model. Heafield [9] presented a KenLM that implemented two data structures, the PROBING data structure and TRIE data structure, for efficient language model queries.

Mathias [10] and Kolak [11] captured the nature language model with a WFSA in speech translation. They used the concept of WFSA to represent the knowledge in a uniform, and broke the complex problems into a cascade of simple WFSA. Nasr [12] introduced the WFSA to construct a new kind of LM by several local models and a general model using a greedy finite state partial parser. Chiang [6] investigated Bayesian inference for WFSA and demonstrated the genericity of this framework which improved performance over EM. Most of current studies reduced the WFSA by a standard operation as minimization. Our WFSA based LM is designed with *trie* structure, which has already stored the LM in a compact way.

The rest of this paper is organized as follows: Sect. 3 introduces the motivation for our approach with the basic concept of LM and WFSA; Sect. 4 describes the system implementation, including the data structure of WFSA based LM and query method. Sect. 5 reports and analyzes the experimental results; and the conclusions are given in Sect. 6.

3 Motivation

3.1 N-gram Language Model

Generally, statistical language model are used to assign probabilities to string of words or tokens. Let w_1^L denote a string of L tokens over a fixed vocabulary. The n-gram language model assigns a probability to w_1^L according to

$$P(w_1^L) = \prod_{i=1}^{L} P(w_i|w_1^{i-1}) \approx \prod_{i=1}^{L} \hat{P}(w_i|w_{i-n+1}^{i-1}) \tag{1}$$

where the approximation reflects a Markov assumption that only the most recent $n-1$ tokens will be considered when predicting the next word.

The smoothing techniques [13,14] are usually introduced with back-off to avoid the sparse data problem in modeling the LM. The context-dependent back-off is used as follows

$$P(w_i|w_{i-n+1}^{i-1}) = \begin{cases} \pi(w_{i-n+1}^{i}) & w_{i-n+1}^{i} \in LM \\ \lambda(w_{i-n+1}^{i-1}) \cdot P(w_{i-n+2}^{i}) & \text{others} \end{cases} \quad (2)$$

where $\pi(\cdot)$ are pre-computed and stored probabilities, and $\lambda(\cdot)$ are back-off weights of the history. The LM file contains the parameter $\pi(\cdot)$ for each listed n-gram, and the parameters $\pi(\cdot)$ and $\lambda(\cdot)$ for each listed m-gram, $1 \leq m < n$; for unlisted m-grams, $\lambda(\cdot) = 1.0$ by definition.

3.2 Query Problems in *N*-gram LM

The state-of-art language model toolkit SRILM [15] uses *trie* to organize the n-grams. Querying in *trie* can be usually composed by two types: forward query and back-off query. However, both of the two query types will be a waste of time as: 1) if the forward query reaches the leaf node of *trie*, it has to do the forward query from the beginning of *trie* when the next word comes in. 2) if the n-gram is not involved in the LM, it has to do the back-off query still from the beginning of *trie*. Thus many nodes in *trie* are traversed with out any use.

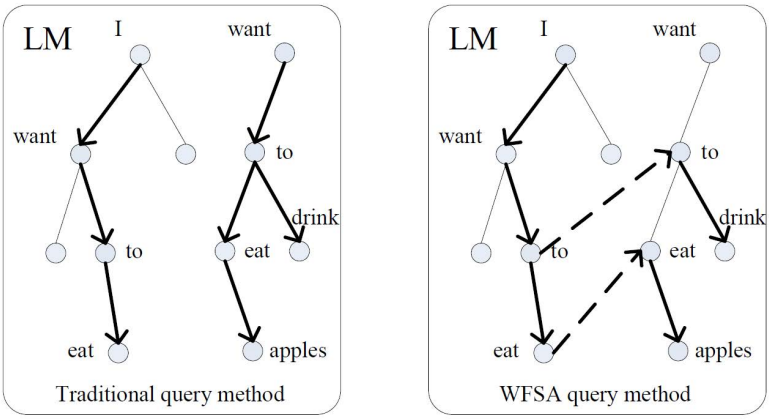

Fig. 1. The comparison of 4-gram LM query methods for "I want to eat apples" and "I want to drink"

A sample of traditional LM query is shown on the left side in Fig.1. The arrows in *trie* represent the query tracks. Although we only need the probability of want to eat apples when calculating the 4-gram LM of "I want to eat apples", the nodes such as "want", "want to" and "want to eat" are traversed completely useless. Moreover, because of the fragment "I want to drink does not involved

in LM, it has to query the sub fragment "want to drink" according to (2). Thus the nodes "want" and "want to" still traversed uselessly.

We do not consider the query in LM as a random procedure but a continuous process. Still take the case in Fig.1 as an example. If we use a roll-back pointer to guide the query directly to an equivalent state [8], the next query can simply proceed by only one search. This gives us the instinct to use a WFSA to solve this problem. Figure 1 at the right side shows the query method in a WFSA based LM. We can see the WFSA will also speed up the back-off queries by the introduction of roll-back pointer. Thus both of the two query problems in LM can be successfully resolved with the concept of WFSA.

4 *N*-gram Language Model Based on WFSA

4.1 WFSA

A WFSA is conceived as an abstract automaton with a finite number of states. The state it is in at any given time is called the current state. It can change from one state to another when initiated by a triggering event or condition which is called the transition, and carry out weights for each transition.

Normally, a typical WFSA can be assigned by a 5-turple $M = (Q, \Sigma, I, F, \delta)$, where Q is a set of states, $I \subseteq Q$ is a set of initial states, $F \subseteq Q$ is a set of final states, Σ is the alphabet which represents the input and output labels, and $\delta \subseteq Q \times (\Sigma \cup \varepsilon)$ is the transition relation. A transition is labeled with ε if it can be traversed without input symbol. The label w_i in a input string L is accepted by the automation M is defined to be

$$L(M) \equiv \{w_i \in \Sigma, q \in Q | \delta(q, w_i) \cap F \neq \emptyset\} \tag{3}$$

The state is traversed according to the input labels until it reaches one of the final states. For each transition step, there will be an output which represents the weighted element.

4.2 WFSA Based LM Structure

We use *trie* as the basic structure of our compact WFSA language model. The nodes in *trie* are the set of finite state Q, and the root of *trie* is the initial state I. Each node of *trie* except the root is the set of final state F. The input label Σ is the alphabet of input sentences, and the weights are the probabilities for n-gram and back-off. The transition relation δ is composed by forward transition T_f and roll-back transition T_b. The forward transition traverses along the path of *trie* which is corresponding to forward query and the roll-back transition traverses with the roll-back pointer which points to the equivalent position in *trie*. It should be noted that the roll-back transition triggers spontaneously without any input when it reaches to the leaves of *trie* or carries out back-off queries, which represents the ε transition in WFSA.

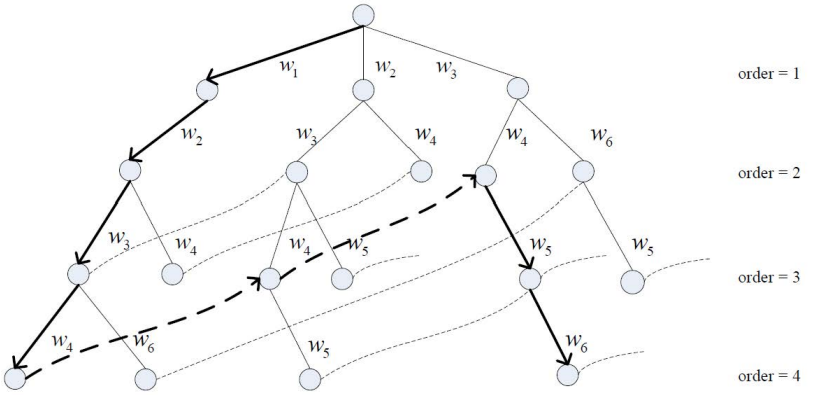

Fig. 2. The structure of a 4-gram WFSA based LM

Figure 2 shows an example of 4-gram LM based on the compact WFSA. Nodes in the *trie* are based on the sorted arrays [16] with probability, back-off, and an index (the solid line in Fig.2) into higher order of n-gram LM. Different with the previous work, the nodes of our WFSA based LM store a roll-back pointer (the dash line in Fig.2) for the m-order ($3 \leq m \leq n$). It is just because of these roll-back pointers that make our LM act as WFSA, which will be illustrated later in the next section. The 2-order omits roll-back pointer to 1-order as it can be easily queried by the vocabulary identifier. The nodes of w_5 at the 3-order and w_6 at the 4-order are pointing to the corresponding equivalents which is not shown in Fig.2. Notice the roll-back pointer in w_6 at the 4-order can safely point to the one at the 2-order, for the back-off probability is restricted to 1.0 if the direct back-off is not involved as described in Sect 2.1.

4.3 Query with WFSA Based LM

For a given input of word sequence w_1^L, the query process of LM can be seen as a series of state transitions based on WFSA. Take the 4-gram language model in Fig.2 as an example. The transition of query state is triggered by each input word $w_i (1 \leq i \leq 6)$ in w_1^6, and each state is corresponding to the node in *trie*. The state transition process is shown in Fig.3. The state s_i^j is represented by the node in LM, and i and j are the beginning and ending identifier of query fragments respectively.

It can be found that each transition triggered with the input of word (or *null*). The forward transition T_f is triggered for each of the input words. If the state is not involved in LM, the query process will continue traversing to the current state and trigger a roll-back transition T_b, until it successfully makes a forward transition. The roll-back transition T_b is just corresponding to the ε transition in WFSA.

The example of LM query process in Fig.3 is processed as follows: After initializing the query process, the current query state firstly traverses to the initial

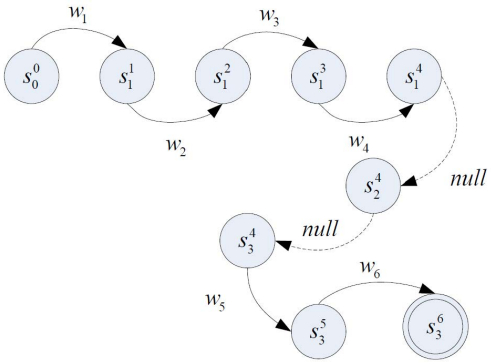

Fig. 3. N-gram LM query based on WFSA

state s_0^0. Then the state traverses forward to s_1^1, for the fragment w_1 exists in the language model. The same situation happens when w_2 and w_3 input, and the current state has reached s_1^3. Then the state transfers to s_1^4 and rolls back to s_2^4 spontaneously according to the roll-back pointer as it has reached the leaf node of *trie*. When word w_5 inputs, the state continues rolling back to s_3^4, as the state s_2^5 does not exist, until it traverses forward to the state s_3^5. At last, the current state gets to the final state s_3^6 and quits the query process after the last word w_6 inputs.

The pseudo code of query WFSA based LM is shown as follows:

Algorithm. Query WFSA based LM

```
Input: word string w_1^L
Output: LMscore of w_1^L
for i = 1 to L do
    while(1)
        if w_{i-n+1}^i ∈ LM
            LMscore *= π(w_{i-n+1}^i) and break
        else
            LMscore *= λ(w_{i-n+1}^i)
        end while
    end for
return LMscore
```

4.4 Hash Cache

To make further speed up of queries, we add a simple hash cache to our WFSA based LM to cache the repetitive queries when decoding a sentence in SMT system. Our cache uses an array of key-value pairs with the size fixed to 2^b for some integer b(we used 24). We choose the current state and input word as the key of our hash table. We use a b-bit hash function to compute the address in

an array where we will place an output state and the carried out probability for each state transition.

For each query of the cache, we check the address of a key given by the b-bits hash. If the key located in the cache array matches the query key, then we get the output state and probability which are stored in cache. Otherwise, we fetch the probability from LM with WFSA and place the new key and value in the cache. The cache will be cleared for each of the translated sentences.

Both of the forward and back-off query speed can be improved by the cache as there are many repetitive LM queries in SMT. Moreover, when calculating the probability for an n-gram which is not involved in the LM, the back-off must perform multiple queries to fetch the necessary back-off information, although the WFSA based LM query has already improved the query speed substantially. Our cache retains the fully results of these calculations and thus saves additional computations in back-off queries.

5 Experiments and Results

5.1 Setup

We used the state-of-art hierarchical phrase-based translation system [17] as our baseline, and test the LM query efficiency on two Chinese-to-English translation tasks: IWSLT-07 (dialogue domain) and NIST-06 (news domain). The test sets of the two domains are IWSLT-07 test sets and NIST-06 test sets, which contain 489 sentences and 1664 sentences separately. We obtained the translation models (TM) following the same constraints as in Chiang [18]. We trained the 4-gram and 5-gram language models using SRILM and then converted them to our WFSA structure before decoding. The training corpora of the experiments are listed in Table 1.

Table 1. Training corpus for LM and TM

Tasks	Model	Parallel sentences	Chinese words	English words
IWSLT-07	TM[1]	0.38M	3.0M	3.1M
	LM[2]	1.3M	—	15.2M
NIST-06	TM[3]	3.4M	64M	70M
	LM[4]	143.M	—	377M

[1] The parallel corpus of BTEC (Basic Traveling Expression Corpus) and CJK (China-Japan-Korea corpus).
[2] The English corpus of BTEC+CJK+CWMT2008.
[3] LDC2002E18, LDC2002T01, LDC2003E07, LDC2003T17, LDC2004T07, LDC2004T08, LDC2005T06, LDC2005T10, LDC2005T34, LDC2006T04, LDC2007T09.
[4] LDC2007T07.

5.2 Results

Storage Space Experiments. We tested our implementation of WFSA based LM (**WFSA**) on the two tasks. Notice that there are no additional nodes in *trie* structure. For each node, there are only 5 additional bytes comparing with the **SRILM**. The bytes are composed by a 4 bytes integer and a 1 bytes character, which represent the roll-back pointer. Thus, the size of WFSA based LM is corresponding to the node number of *trie*, which makes it compact. The results are shown in Table 2.

Table 2. The comparison of LM size between SRILM and WFSA

Tasks	N-grams	**SRILM**(Mb)	**WFSA**(Mb)	Δ(%)
IWSLT-07	4	65.7	89.1	35.6
	5	89.8	119.5	33.1
NIST-06	4	860.3	1190.4	38.4
	5	998.5	1339.7	34.2

In Table 2, the storage sizes of WFSA based LM increase about 35% than the **SRILM** in the two domains. It is because of the extra bytes that keep the roll-back pointer in the *trie* node. However, the increment is acceptable as it is linearly dependent with the nodes of *trie*.

Query Speed Experiments. We used the 4-gram and 5-gram LM in IWSLT-07 and NIST-06 to test the query efficiency. We measured the time[5] required to perform each query in actual translation using the **SRILM** as our baseline. Then we measured the time of using WFSA based LM (**WFSA**) and introduced our cache to WFSA (**WFSA+cache**) to speed up the query. Times were averaged over 3 runs on the same machine. The results are shown in Table 3.

As expected, the use of **WFSA** speeds up the LM query substantially. The query speed in both domains has been improved by 57.1% and 59.5% in 4-gram LM, and 67.4% and 68.4% in 5-gram LM separately. The improvement of query speed in 5-gram LM is much better than 4-gram by about 10%. It suggests that our implementation of WFSA is suitable for higher n-grams. Although the WFSA has already improved the speed, it can be found our **WFSA+cache** queries more effectively. The query speed can be further improved by the introduction of cache in all of the translation tasks. The final implementation of our **WFSA+cache** can speed up the query by about 75%.

[5] All experiments were performed on a DELL server, with an Intel Xeon 5130 CPU running at 2.00 GHz and 8M of cache. The operation system is Ubuntu 7.10.

Table 3. The comparison of query time with different methods

N-grams	Methods	IWSLT-07(s)	NIST-06(s)
4	**SRILM**	163	15433
	WFSA	70	6251
	WFSA+cache	42	3907
5	**SRILM**	261	25172
	WFSA	85	7944
	WFSA+cache	59	6128

Analysis. We measured the probability of repetitive queries and back-off queries in 4-gram LM that occurred during decoding, which had a close relationship with our WFSA based LM. The results on the two tasks are shown in Table 4.

Table 4. The probability of back-off and repetitive query in 4-gram LM

Tasks	Back-off	Repetitive
IWSLT-07	60.5%	95.5%
NIST-06	60.3%	96.4%

It can be seen that back-off queries are widely existed in statistical machine translation and as many as about 60% in 4-gram queries are proceeding for back-off query. Notice that the probabilities of back-off queries are similar in both tasks, which are corresponding to the increment of query efficiency. It suggests that our model can effectively accelerate the back-off queries.

Although decoding a single sentence can trigger a huge number of LM queries, it can be found most of these queries are repetitive. Therefore, keeping the results of LM queries in a cache can be effective at reducing overall queries. This has been confirmed by our experimental results in the query speed experiments above.

6 Conclusions

We have presented a new method for faster LM implementation based on compact WFSA. We consider the LM query process as a series of state transitions with WFSA according to the context character of LM. This method improves the query speed effectively with a compact LM in size for SMT system. We have also described a simple caching technique which leads to performance improvements in over all decoding time. Our WFSA based language model can not only be used in the faster query of LM in SMT, but also be suitable in other nature language processing, such as speech recognition and information retrieval, etc.

References

1. Thorsten, B., Popat, A.C., Peng, X., Franz, J.O., Jeffrey, D.: Large Language Models in Machine Translation. In: Proceedings of EMNLP-CoNLL, pp. 858–867 (2007)
2. Goodman, J.: A Bit of Progress in Language Modeling. Technical report. Microsoft Research (2001)
3. Marcello, F., Mauro, C.: Efficient handling of n-gram language models for statis tical machine translation. In: Proceedings of the 2nd Workshop on Statistical Machine Translation, pp. 88–95 (2007)
4. David, T., Miles, O.: Randomised language modelling for statistical machine translation. In: Proceedings of the ACL, pp. 512–519 (2007)
5. Kevin, K., Jonathan, G.: An overview of probabilistic tree transducers for na tural language processing. In: Proceedings of CICLing (2005)
6. David, C., Jonathan, G., Kevin, K., Adam, P., Sujith, R.: Bayesian inference for Finite-State transducers. In: Proceedings of the NAACL, pp. 447–455 (2010)
7. Adam, P., Dan, K.: Faster and Smaller N-Gram Language Models. In: Proceedings of the ACL, pp. 258–267 (2011)
8. Zhifei, L., Sanjeev, K.: A scalable decoder for parsing- based machine translation with equivalent language model state maintenance. In: Proceedings of the Second Workshop on Syntax and Structure in Statistical Translation, pp. 10–18 (2008)
9. Kenneth, H.: KenLM: Faster and Smaller Language Model Queries. In: Proceedings of the 6th Workshop on Statistical Machine Translation, pp. 187–197 (2011)
10. Lambert, M., William, B.: Statistical phrase-based speech translation. In: Proceedings of ICASSP (2006)
11. Okan, K., Willian, B., Philip, R.: A generative probabilistic OCR model for NLP applications. In: Proceedings of the HLT-NAACL (2003)
12. Alexis, N., Yannick, E., Frédéric, B., Thierry, S., de Renato, M.: A language model combining N-grams and stochastic finite state automata. In: Proceedings of Eurospeech (1999)
13. Reinhard, K., Hermann, N.: Improved backing-off for m-gram language modeling. In: Proceedings of the IEEE International Conference on Acoustics, Speech, and Signal Processing, pp. 181–184 (1995)
14. Slava, M.K.: Estimation of probabilities from sparse data for the language model component of a speech recognizer. IEEE Transactions on Acoustics, Speech and Signal Processing, 400–401 (1987)
15. Andreas, S.: SRILM: An extensible language modeling toolkit. In: Proceedings of Interspeech (2002)
16. Edward, W., Bhiksha, R.: Quantization based language model compression. In: Proceedings of Eurospeech (2001)
17. David, C.: A hierarchical phrase-based model for statistical machine translation. In: Proceedings of ACL, pp. 263–270 (2005)
18. David, C.: Hierarchical phrase-based translation. Computational Linguistics 33(2), 201–228 (2007)