LEXICALIZED REORDERING IN MULTIPLE-GRAPH BASED STATISTICAL MACHINE TRANSLATION

Bowen Zhou, Rong Zhang and Yuqing Gao

IBM T. J. Watson Research Center
Yorktown Heights, NY 10598
Email: {zhou, zhangr, yuqing}@us.ibm.com

ABSTRACT

Reordering is one of the key problems in statistical machine translation (SMT). This paper first presents how lexicalized reordering is embedded into a phrase-based SMT framework modeled by multiple-graph that was formulated in our previous work. Specifically, we show how lazy reordering graph is computed and combined with our previously proposed multiple layer search (MLS) to achieve an efficient reordering decoding that is also flexible to take various reordering models. Secondly, we introduce a variety of lexicalized reordering models employed in our system that significantly improved system performance.

Index Terms— speech translation, multiple-graph decoding, lexicalized reordering, finite automata, machine translation

1. INTRODUCTION

Phrase-based statistical machine translation (SMT) has been widely applied in many state-of-the-art speech translation systems, as significant progresses have been made during past years. One major reason for the success is that the explicit usage of phrases in modelings helps for both better word sense disambiguition and local word orderings in translation. However, how to construct sensible long-distance orderings of a target translation at a phrase level, particularly for language pairs that differ substantially in syntax orders, remains to be a major challenge for such phrase-based SMT systems. The challenge comes from two perspectives. First, reliable modeling of reorders itself is difficult mainly due to the data sparseness to consider an exponential number of possible orderings; Secondly, arbitrary reordering makes the translation decoding a NP-complete problem [1], which results in more complexities in designing a decoder and the decoder is more liable to search errors. In the literature, reordering in SMT has been and is currently extensively studied by many researchers, for example in [2, 3, 4, 5, 6], to name just a few.

In our previous works, we proposed the Folsom SMT system [7, 8], which formulates the phrase-based SMT using multiple-graph with offline finite state optimization. Among these graphs, two graphs are built offline. One of them encodes entire phrasal level translation model; and the other represents target language model. There is another graph that is expanded on-the-fly during decoding to represent source input. In this framework, the decoding procedure is handled by the proposed Multiple Layer Search (MLS) algorithm, which can be viewed as an optimized version of a Viterbi search combined with lazy compositions of all these graphs simultaneously, in addition to embedded dynamic minimization. Our method shows significant speed and memory advantages [7] over more commonly used stack decoders in addition to the flexibility support of using arbitrary N-gram language model in decoder, which enables Folsom to be a converged SMT system for scalable platforms including PDA's under limited computational resources. Moreover, due to the usage of graph and the MLS search, this framework possesses additional advantages of allowing more effective integration of speech recognition lattice output with the SMT graphs to achieve improved speech translation accuracy [8].

In this paper, we particularly discuss how to address reordering issues for both modeling and effective decoding in such a multiple-graph based SMT system. Since reordering complexity is exponential, building a full reordering graph statically is expensive and unnecessary as most of permutation paths in the graph will not eventually lead to any sensible translation results. Therefore, we instead compute the reordering graph in a lazy fashion, and the reordering states are composed with translation and language model states dynamically during search, which allows us to perform an effective and virtually loss-free pruning using joint scores from all models at an early stage to reduce search space dramatically. The first part of this paper presents how the input graph is dynamically extended to account for all considered source side reorderings based on some given reordering models; we will show why this implementation combined with MLS algorithm allows for flexible reordering models without modifying the core part of decoder. In following, we propose a number of lexicalized reordering model training algorithms to estimate reorderings at both word and phrase levels, with a particular focus on a data-driven clustering algorithm that is employed to alleviate the issues of data sparseness.

The remaining part of this paper is organized as follows: Sec. 2 starts with an overview of our multiple-graph based SMT system, followed by the decoding algorithm that handles flexible reorderings; Sec. 3 describes details of our lexicalized reordering models; Sec. 4 presents experimental results for speech translation tasks; and finally, Sec. 5 summarizes our contributions.

2. LAZY REORDERING AND MLS DECODING FOR MULTIPLE-GRAPH BASED SMT

2.1. Multiple-Graph based SMT

Phrase-based SMT systems utilize parallel corpus with unsupervised word-level alignments to learn phrasal translation models. The source input, assuming a foreign word sequence, f_1^J is thus segmented into acceptable phrase sequences $\bar{f}_1^K$, where $1 \leq K \leq J$, to search for the best target English word sequence e_1^I by combining various phrasal level translations.

In our multiple-graph based system, we express the entire trans-

We thank the DARPA TransTac program for providing partial funding for this work.

lation procedure as a chain of conditional probabilities, which can be represented by finite-state machines (FSM's) that model the relationships between inputs and outputs [7]. Therefore, the translation task can be framed as finding the best path in the following FSM:

$$\mathcal{S} = \mathcal{I} \circ \mathcal{P} \circ \mathcal{T} \circ \mathcal{W} \circ \mathcal{L} \quad (1)$$

where '$\circ$' is the composition operator, $\mathcal{I}$ denotes the source word sequence expressed as a finite-state automaton (presented in details in Sec. 2.2), and the transducers $\mathcal{P}$, $\mathcal{T}$, $\mathcal{W}$, and $\mathcal{L}$ correspond to source language segmentation transducer, phrase translation transducer, target language phrase-to-word transducer, and target language model, respectively.

To minimize the amount of computation required at decoding time, it is desirable to perform as many composition operations in Eq. (1) as possible ahead of time. While it may not be feasible to compute $\mathcal{H} = \mathcal{P} \circ \mathcal{T} \circ \mathcal{W} \circ \mathcal{L}$ in its entirety as a single FSM, it *is* possible to separate $\mathcal{H}$ into two pieces: the language model $\mathcal{L}$ and the translation model $\mathcal{M}$, which is a log-linear combination of a number of sub-translation models.

$$\mathcal{M} = Min(Min(Det(\mathcal{P}) \circ \mathcal{T}) \circ \mathcal{W}) \quad (2)$$

where *Min* denotes the minimization operation. Due to the determinizability of $\mathcal{P}$ [7], $\mathcal{M}$ can be computed offline using a moderate amount of memory.

Therefore, our translation system is built upon on three graphs $\mathcal{I}$, $\mathcal{M}$ and $\mathcal{L}$, and the decoding problem is formulated as finding the best path in the following dynamically-composed graph: $\mathcal{I} \circ \mathcal{M} \circ \mathcal{L}$.

To address the problem of efficient search, we have developed a multiple layer search algorithm. More details of the MLS are presented in [7, 8], and we provide an overview here. The search initializes with a joint beginning state $\vec{s}_0$ with no accumulated cost, which includes the initial state at each layer. Starting from a state traversal in $\mathcal{I}$, the active states in $\mathcal{M}$ are expanded by matching the arc labels that are consumed in $\mathcal{I}$; and next, active states in layer $\mathcal{L}$ are expanded in a similar fashion to match the output of $\mathcal{M}$. The costs from each layer are combined together with heuristic scores such as word counts and phrase counts penalties using log-linear models. The procedure is repeated until a joint final states $\vec{s}_e$ is achieved. The set of legal translation candidates are those where each component substate is a final state in its layer. The selected candidate is the legal candidate with the lowest accumulated cost.

It should be noted that we update language model scores at every target word by traversing $\mathcal{L}$, and that our decoder is flexible to take an arbitrary-order of N-gram LM in a graph.

2.2. Reordering Search in Multiple-Graph based SMT

During decoding, we assume that target sentence is produced left-to-right, and all reordering is conducted on the source side. Every state in the source input graph $\mathcal{I}$ represents the source side coverage status, and this requires $O(2^J)$ states in a static graph to represent all possible permutations of the source side, which is computationally intractable. Thus, $\mathcal{I}$ is computed on-the-fly, subject to certain reordering models and constraints. The lazy nature of reordering graph computation is similar in spirit to [4], where distance-based reordering models are used. Within our multiple-graph based framework, representing reorderings as another layer naturally fits. Moreover, the lazy computation coupled with the MLS decoding scheme provides an effective way to conduct beam pruning for both reordering graph and joint hypotheses combining all three layers. In following, we will particularly show how lexicalized reordering and decoding can be implemented in the graph.

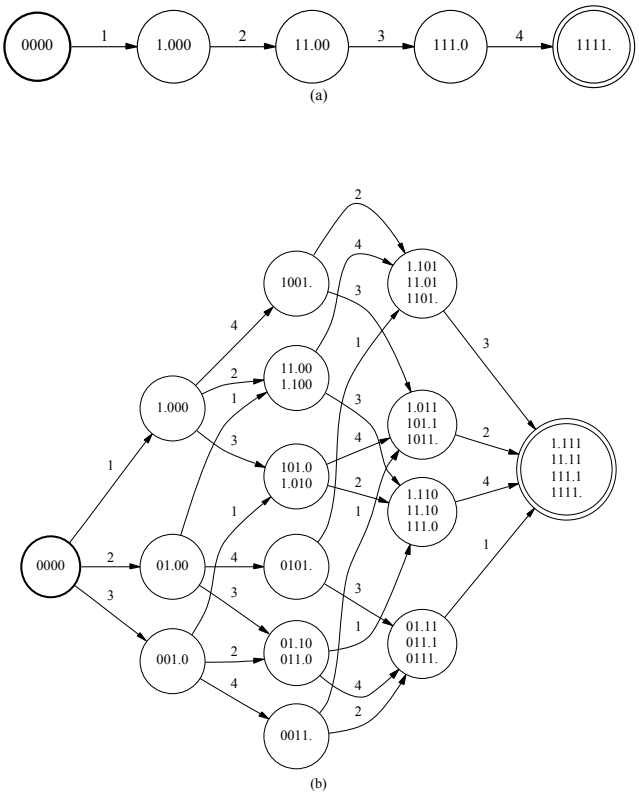

Fig. 1. Input graph computed on-the-fly during search. (a): monotone decoding is a linear left-to-right automata; (b): reorder graph with the IBM constraint of skips less than three. Costs associated with each arc are omitted here and they are determined by specific reordering models used by the decoder.

Each state in $\mathcal{I}$ denotes a specific input coverage, which is represented by a bit vector. The initial state leaves all bits zero and the final state has all bits one. Arcs are labeled with an input position to be covered linking start and end states. Arcs and their weights are determined by the specific reordering models being used by the decoder. Every path from the initial state to final state represents an acceptable permutation of the input.

It should be noted that the MLS decoding procedure is independent from the topology of $\mathcal{I}$. Therefore, our scheme is desirable to achieve a modularized SMT engine, where a unified core decoder implementation is detached from reordering modules. Various reordering constraints and models can be plugged-in immediately by providing a specific graph expanding function, needless to modify the core part of the decoder.

For example, monotone decoding can be regarded as a special case, where source and target languages are assumed to share the same order at phrase level. The input layer $\mathcal{I}$ is thus constructed as a trivial left-to-right FSA, which contains only one deterministic path from the initial state to end state, as shown in Fig. 1(a).

For more general cases, every state in the input graph is expanded by covering a yet uncovered position up to this state, and states with same bitvector are merged. However, we need to further distinguish the last covered positions within the same state in order to calculate following reordering costs properly. Therefore, we track

the last covered position with a "dot" immediately following the "1" of that position. For example, an arc outgoing from state "1001." with label 3 will merge into state "101.1". As we pool all the reordering states sharing the same coverage bitvector together, we are able to perform more efficient beam pruning during the lazy graph expansion, independent of the positions of last covered bit.

As mentioned above, the unified interface for reordering graph construction enables our system to support arbitrary reorderings. In practice, however, unconstrained reordering probabilities are difficult to estimate reliably, and we need to reduce search space for speed and memory concerns. Thus, we typically construct the reordering graphs subject to the IBM constraint [9], i.e., only up to k first positions in the input are allowed to be left uncovered in expanding one state to another. In addition, we also define a maximum window size that is the distance between the first uncovered position and the right-most covered position. Fig. 1(b) shows a reordering graph computed for a four-word input with a maximum skip of 3. Note that arc costs are omitted for simplicity.

On top of certain reordering constraints, the topology and associated arc weights are also determined by reordering models, which is described in Sec. 3. The joint decoding procedure combining three layers using a MLS algorithm is identical to the one we presented in [8], with reordering costs accumulated along the arcs in $\mathcal{I}$.

3. ESTIMATING LEXICALIZED REORDERING MODELS

A widely-used reordering models is based on relative distances [10] where higher degree of non-monotonic jumps are consistently punished independent of actual words or phrases being considered. In our system, we utilize the lexicalized reordering models that condition the reordering probability not only on relative distances but also on the actual words. In Sec. 3.1 we first describe two models based on word and phrase distortion, and propose improved models to handle data sparseness issues in Sec. 3.2.

3.1. Lexicalized Reorder Models: P- and Q- Model

Our first lexicalized reordering model, denoted as P-model, is similar to the out-bound model described in [6]. When expanding from one state with the last covered word being f_j to another state following an arc labeled with input word f_i, we want to estimate the probability of relative jump $d = i - j$ as $P(d|f_j)$. Such a distribution can be directly estimated from word-aligned parallel corpus. Assuming f_j is aligned to e_m, we have:

$$\hat{P}_P(d|f_j) = \frac{C(f_j, d)}{\sum_{d_k} C(f_j, d_k)} \tag{3}$$

where $C(f_j, d_k)$ is the count of e_{m+1} is being aligned to f_{j+d_k} from entire training data. In practice, the estimated distribution is further smoothed by interpolating with a distance-decreasing distribution to account for over-training issues. To apply this model to a phrase based system, we apply such distortion probabilities directly for words at a phrase boundary, and discount the probability for words interior of a phrase.

The next model, denoted as Q-model, takes into account the phrase alignments obtained from training data. First, we force-align source sentence f^l with target sentence e^l at phrase level based on phrasal translation models. The forced alignment is obtained through a search procedure that maximizes the following objective function:

$$\theta^l = \prod_k P(\bar{e}_k|\bar{f}_{a(k)}), s.t. \bigcup_k \bar{e}_k = e^l, \bigcup_{a(k)} \bar{f}_{a(k)} = f^l \tag{4}$$

where $a(k)$ is the aligned phrase position. We consider a maximum jump window such that $|k - a(k)| < 10$. For each sentence l, we extract top ten phrase pair alignments. Next, we reposition phrases in f^l per the order of their counterparts in e^l. Suppose f^l_j is the ending word of a phrase and f^l_k is the beginning word of the following phrase after repositioning, we calculate phrase boundary distortion probabilities for f_j as follows:

$$\hat{P}_Q(d|f_j) = \frac{\sum_l C(f^l_j, f^l_k)}{\sum_{f_k, l} C(f^l_j, f^l_k)}, d = k - j \tag{5}$$

For words only observed interior of a phrase, we set $\hat{P}_Q(d|f_j) = 1$ for $d = 1$ and 0 elsewhere. The model is then smoothed in a same fashion as the P-model.

3.2. Models: CQ-Model

Lexicalized reordering models paint a much more refined picture for the ordering options during search. However, the distribution estimation may suffer from data sparseness issue. Often, most distortion positions for many words are not observed in the training data, thus the model relies on the smoothing component, which falls back to a distance-based model.

The data sparseness issue could be alleviated to some extent by clustering words. There are at least two schemes to achieve a clustering. First, one can group words based on their part-of-speech tags as they tend to share same reordering patterns, such as 'NP ADJ' often rewrites as 'ADJ NP' in translating Arabic into English. However, tag-annotated data is required to train a reliable automatic tagging model. If such an annotation is not available, an alternative is to perform an unsupervised data-driven clustering.

We propose such a data-driven clustering procedure that is built upon P- and Q-models to group words sharing similar reordering patterns in translation. First, we utilize a vector to describe the word's specific distortion characteristics, that is, every word w is regarded as such a point in a high-dimensional space:

$$\vec{D}(w) = (\ldots, \hat{P}_P(d|w), \ldots, pe), d \neq 0 \tag{6}$$

where pe is the probability that w appears at a phrase end, which is estimated from forced phrase alignments.

Based on Q-model, we assign every word into one of two groups depending on how often it appears at a phrase end. Next, each group is further divided into one of following three classes, left-jump, right-jump or no-jump, based on each word's P-model probability. Next, a bottom-up clustering is conducted for each sub-group, using the average linkage clustering based on $\vec{D}(w)$. For obtained cluster G, the shared distortion probability of $f_j \in G$ is defined as:

$$\hat{P}_{CQ}(d|f_j) = \frac{\sum_{f_j \in G} P_Q(d|f_j) \times occ(f_j)}{\sum_{f_j \in G} occ(f_j)} \tag{7}$$

4. EXPERIMENTAL EVALUATION

4.1. Experimental Setup

To evaluate our system described in this paper, experiments are carried out for a speech translation task translating Farsi into English, as part of the evaluation tasks of DARPA TransTac program. The parallel training data has a total of 110K parallel sentences. For our system development, we held out the data randomly selected from the available parallel corpus as the internal dev and test set, each of

Table 1. Data description for training, dev and test sets.

Data	Farsi	English
Training (words)	960K	1.0M
Vocabulary size	26K	14K
Dev set (1 ref.)	1430 sent.	
Running words	11505	13059
Test set 1 (1 ref.)	1390 sent.	
Running words	10353	11894
Test set 2 (4 refs.)	416 sent.	
Running words	4158	–

Table 2. SMT performance (BLEU %) of various reordering models.

System	Dev-1ref	Test1-1ref	Test2-4ref
Monotone	36.97	28.95	26.39
Lexicalized Reordering			
P (w=5, s=3)	38.73	31.67	30.09
Q (w=5, s=3)	38.85	31.86	31.53
CQ (w=5, s=3)	38.91	31.78	31.80
P (w=8, s=3)	38.82	32.02	30.77
Q (w=8, s=3)	39.22	31.72	31.69
CQ (w=8, s=3)	39.45	31.98	31.85

them comes with 1 reference. In addition, we evaluate our proposed system using DARPA's official offline evaluation set, which includes 4 references. Data description is summarized in Table 1.

As the focus of this paper is the lexicalized reordering graph $\mathcal{I}$, we follow a more or less standard procedure here to build our translation model. The parallel corpus is first aligned in both directions with GIZA++ [11], and phrase pairs are extracted from the heuristic combination of the both alignments. The maximum phrase length was set to 8 on source side, and thus, we obtained over 1M phrase pairs from the training data. The resulting translation graph $\mathcal{M}$ has about 4M states and 5M arcs. The language model graph $\mathcal{L}$ is computed from a back-off 4-gram with modified Kneser-Ney smoothing, which is trained from the English side of the parallel corpus.

4.2. Experimental Results

The reordering graph $\mathcal{I}$ is computed on-the-fly for all experiments. All feature weights are optimized on dev set with a line search method. First, we set the baseline as the monotone models, i.e., no reordering above phrase level is allowed. Next, we plug in various lexicalized reordering models discussed above. Under the IBM constraint, we set the maximum skip $s = 3$, and report experimental results with maximum reorder window $w = 5$ and $w = 8$. Results are summarized in Table 2.

We observe that all lexicalized reordering systems achieve significant gains over monotone, particularly on Test2 with 4 references, ranging from 3.7% to 5.4% absolute BLEU improvements. Secondly, increasing the reordering window size from 5 to 8 provides a generally better performance, though it is not always significant or consistent.

Comparing different lexicalized reordering models, Q-model outperforms slightly over P-model. This could be explained by that the former has more tight correlations with the phrasal translation model, as it directly models outbound probabilities of phrase end words. For the CQ-model, we cluster all Farsi words in about 700 classes, with about 200 frequent words remain in their own exclusive class. The experimental results show that CQ-model achieves marginal but mostly consistent gains over Q-model. We speculate that the impacts of clustering could have already been captured to some extent by the smoothing procedure used for Q-model. Overall, the best performance over all test sets is obtained by using the CQ-model and a larger reordering window.

5. SUMMARY

In this paper, we presented how lexicalized reordering models are applied in a multiple-graph based SMT system. The reordering is modeled by a graph during decoding that is dynamically expanded with respect to specified reordering models. We showed that combined with our previously proposed Multiple-Layer Search algorithm, the on-the-fly reordering graph computation enables us to employ flexible reordering models, needless to modify the core decoder. Furthermore, we proposed a number of data-driven lexicalized reordering models that we argue significantly improved system performance over baseline monotone. Specifically, a clustered reorder model shows some potential to help further in the presence of sparse data.

6. REFERENCES

[1] K. Knight, "Decoding complexity in word-replacement translation models," *Computational Linguistics*, vol. 25, no. 4, pp. 607–615, 1999.

[2] D. Wu, "Stochastic inversion transduction grammars and bilingual parsing of parallel corpora," *Comput. Linguist.*, vol. 23, no. 3, pp. 377–403, 1997.

[3] C. Tillmann and T. Zhang, "A localized prediction model for statistical machine translation," in *Proc. of the 43rd ACL*, 2005, pp. 557–564.

[4] S. Kanthak, D. Vilar, E. Matusov, R. Zens, and H. Ney, "Novel reordering approaches in phrase-based statistical machine translation," in *ACL 2005 Workshop on Building and Using Parallel Texts: Data-Driven Machine Translation and Beyond*, Ann Arbor, Michigan, June 2005.

[5] S. Kumar and W. Byrne, "Local phrase reordering models for statistical machine translation," in *Proc. HLT/EMNLP*, 2005, pp. 161–168.

[6] Y. Al-Onaizan and K. Papineni, "Distortion models for statistical machine translation," in *Proc. of the 44th ACL*, 2006, pp. 529–536.

[7] B. Zhou, S. F. Chen, and Y. Gao, "Folsom: A fast and memory-efficient phrase-based approach to statistical machine translation," in *IEEE/ACL Workshop on Spoken Language Technology*, 2006.

[8] B. Zhou, L. Besacier, and Y. Gao, "On efficient coupling of asr and smt for speech translation," in *Proc. ICASSP*, 2007.

[9] A. Berger, P. Brown, S. A. Della Pietra, V. J. Della Pietra, A. Kehler, and R. Mercer, "Language translation apparatus and method using context-based translation models," *US Patent*, , no. 5,510,981.

[10] P. Koehn, F. Och, and D. Marcu, "Statistical phrase-based translation," in *Proc. NAACL/HLT*, 2003.

[11] F. J. Och, C. Tillmann, and H. Ney, "Improved alignment models for statistical machine translation," in *Proc. EMNLP/VLC'99*, MD, USA, 1999, pp. 20–28.