Advancements in Reordering Models for Statistical Machine Translation

Minwei Feng and Jan-Thorsten Peter and Hermann Ney
Human Language Technology and Pattern Recognition
Computer Science Department, RWTH Aachen University, Aachen, Germany
<surname>@cs.rwth-aachen.de

Abstract

In this paper, we propose a novel reordering model based on sequence labeling techniques. Our model converts the reordering problem into a sequence labeling problem, i.e., a tagging task. Results on five Chinese-English NIST tasks show that our model improves the baseline system by 1.32 BLEU and 1.53 TER on average. Results of comparative study with other seven widely used reordering models will also be reported.

1 Introduction

The systematic word order difference between two languages poses a challenge for current statistical machine translation (SMT) systems. The system has to decide in which order to translate the given source words. This problem is known as the reordering problem. As shown in (Knight, 1999), if arbitrary reordering is allowed, the search problem is NP-hard.

Many ideas have been proposed to address the reordering problem. Within the phrase-based SMT framework there are mainly three stages where improved reordering could be integrated:

In the preprocessing: the source sentence is reordered by heuristics, so that the word order of source and target sentences is similar. (Wang et al., 2007) use manually designed rules to reorder parse trees of the source sentences. Based on shallow syntax, (Zhang et al., 2007) use rules to reorder the source sentences on the chunk level and provide a source-reordering lattice instead of a single reordered source sentence as input to the SMT system. Designing rules to reorder the source sentence is conceptually clear and usually easy to implement. In this way, syntax information can be incorporated into phrase-based SMT systems. However, one disadvantage is that the reliability of the rules is often language pair dependent.

In the decoder: we can add constraints or models into the decoder to reward good reordering options or penalize bad ones. For reordering constraints, early work includes ITG constraints (Wu, 1995) and IBM constraints (Berger et al., 1996). (Zens and Ney, 2003) did comparative study over different reordering constraints. This paper focuses on reordering models. For reordering models, we can further roughly divide the existing methods into three genres:

- The reordering is a classification problem. The classifier will make decision on next phrase’s relative position with current phrase. The classifier can be trained with maximum likelihood like Moses lexicalized reordering (Koehn et al., 2007) and hierarchical lexicalized reordering model (Galley and Manning, 2008) or be trained under maximum entropy framework (Zens and Ney, 2006).
- The reordering is a decoding order problem. (Mariño et al., 2006) present a translation model that constitutes a language model of a sort of bilanguage composed of bilingual units. From the reordering point of view, the idea is that the correct reordering is a suitable order of translation units. (Feng et al., 2010) present a simpler version of (Mariño et al., 2006)'s model which utilize only source words to model the decoding order.
- The reordering can be solved by outside heuristics. We can put human knowledge into the decoder. For example, the simple jump model using linear distance tells the decoder that usually the long range reordering should be avoided. (Cherry, 2008) uses information from dependency trees to make the decoding process keep syntactic cohesion. (Feng et al., 2012) present a method that utilizes predicate-argument structures from semantic role labeling results as soft constraints.

In the reranking framework: in principle, all
the models in previous category can be used in
the reranking framework, because in the rerank-
ing we have all the information (source and tar-
get words/phrases, alignment) about the transla-
tion process. (Och et al., 2004) describe the use of
syntactic features in the rescoring step. However,
they report the syntactic features contribute very
small gains. One disadvantage of carrying out re-
ordering in reranking is the representativeness of
the N-best list is often a question mark.

In this paper, we propose a novel tagging style
reordering model which is under the category
“The reordering is a decoding order problem”.
Our model converts the decoding order problem
into a sequence labeling problem, i.e. a tagging
task. The remainder of this paper is organized
as follows: Section 2 introduces the basement
of this research: the principle of statistical ma-
chine translation. Section 3 describes the proposed
model. Section 4 briefly describes several reordering
models with which we compare our method.
Section 5 provides the experimental configuration
and results. Conclusion will be given in Section 6.

2 Translation System Overview

In statistical machine translation, we are given a
source language sentence $f_1^J = f_1 \ldots f_j \ldots f_J$.
The objective is to translate the source into a tar-
get language sentence $e_1^J = e_1 \ldots e_i \ldots e_J$. The
strategy is to choose the target sentence with the
highest probability among all others:

$$
\hat{e}_i^J = \arg \max_{I,e_i^J} \{ Pr(e_i^J|f_i^J) \}
$$

(1)

We model $Pr(e_i^J|f_i^J)$ directly using a log-linear
combination of several models (Och and Ney,
2002):

$$
Pr(e_i^J|f_i^J) = \frac{\exp \left( \sum_{m=1}^{M} \lambda_m h_m(e_i^J, f_i^J) \right)}{\sum_{i',e_i'^J} \exp \left( \sum_{m=1}^{M} \lambda_m h_m(e_i'^J, f_i^J) \right)}
$$

(2)

The denominator is to make the $Pr(e_i^J|f_i^J)$ to be a
probability distribution and it depends only on the
source sentence $f_i^J$. For search, the decision rule
is simply:

$$
\hat{e}_i^J = \arg \max_{I,e_i^J} \left\{ \sum_{m=1}^{M} \lambda_m h_m(e_i^J, f_i^J) \right\}
$$

(3)

The model scaling factors $\lambda_i^J$ are trained with
Minimum Error Rate Training (MERT). In this pa-
ter, the phrase-based machine translation system
is utilized (Och et al., 1999; Zens et al., 2002;
Koehn et al., 2003).

3 Tagging-style Reordering Model

In this section, we describe the proposed novel
model. First we will describe the training process.
Then we explain how to use the model in the de-
coder.

3.1 Modeling

Figure 1 shows the modeling steps. The first step
is word alignment training. Figure 1(a) is an ex-
ample after GIZA++ training. If we regard this
alignment as a translation result, i.e. given the
source sentence $f_1^J$, the system translates it into
the target sentence $e_1^J$, then the alignment link set
$\{a_1 = 3, a_3 = 2, a_4 = 4, a_5 = 5, a_6 = 7, a_7 = 6\}$ reveals the decoding process, i.e.
the alignment implies the order in which the source
words should be translated, e.g. the first generated
target word $e_1$ has no alignment, we can regard it
as a translation from a NULL source word; then
the second generated target word $e_2$ is translated
from $f_3$. We reorder the source side of the align-
ment to get Figure 1(b). Figure 1(b) implies the
source sentence decoding sequence information,
which is depicted in Figure 1(c). Using this ex-
ample we describe the strategies we used for special
cases in the transformation from Figure 1(b) to
Figure 1(c):

- ignore the unaligned target word, e.g. $e_1$
- the unaligned source word should follow its
  preceding word, the unaligned feature is kept
  with a * symbol, e.g. $f_2^J$ is after $f_1$
- when one source word is aligned to multi-
  ple target words, only keep the alignment
  that links the source word to the first target word,
  e.g. $f_4$ is linked to $e_5$ and $e_6$, only $f_4 - e_5$
  is kept. In other words, we use this strategy
to guarantee that every source word appears
only once in the source decoding sequence.
- when multiple source words are aligned to
  one target word, put together the source
  words according to their original relative po-
  sitions, e.g. $e_6$ is linked to $f_6$ and $f_7$. So in
  the decoding sequence, $f_6$ is before $f_7$.

Now Figure 1(c) shows the original source sen-
tence and its decoding sequence. By using the
strategies above, it is guaranteed that the source
sentence and its decoding sequence have the ex-
labeled after \( f_j \) and \( f_j \) translated after \( f_{j+1} \).

Up to this point, we have converted the reordering problem into a tagging problem with nine tags. The transformation in Figure 1 is conducted for all the sentence pairs in the bilingual training corpus. After that, we have built an “annotated” corpus for the training. For this supervised learning task, we choose the approach conditional random fields (CRFs) (Lafferty et al., 2001; Sutton and McCallum, 2006; Lavergne et al., 2010) and recurrent neural network (RNN) (Elman, 1990; Jordan, 1990; Lang et al., 1990).

For the first method, we adopt the linear-chain CRFs. However, even for the simple linear-chain CRFs, the complexity of learning and inference grows quadratically with respect to the number of output labels and the amount of structural features which are with regard to adjacent pairs of labels. Hence, to make the computational cost as low as possible, two measures have been taken. Firstly, as described above we reduce the number of tags to nine. Secondly, we add source sentence part-of-speech (POS) tags to the input. For features with window size one to three, both source words and its POS tags are used. For features with window size four and five, only POS tags are used.

As the second method, we use recurrent neural network (RNN). RNN is closely related with Multilayer Perceptrons (MLP) (Rumelhart et al., 1986), but the output of one ore more hidden layers is reused as additional inputs for the network in the next time step. This structure allows the RNN to learn whole sequences without restricting itself to a fixed input window. A plain RNN has only access to the previous events in the input sequence. Hence we adopt the bidirectional RNN (BRNN) (Schuster and Paliwal, 1997) which reads the input sequence from both directions before making the prediction. The long short-term memory (LSTM) (Hochreiter and Schmidhuber, 1997) is applied to
counter the effects that long distance dependencies are hard to learn with gradient descent. This is often referred to as vanishing gradient problem (Bengio et al., 1994).

3.2 Decoding

Once the model training is finished, we make inference on develop and test corpora which means that we get the labels of the source sentences that need to be translated. In the decoder, we add a new model which checks the labeling consistency when scoring an extended state. During the search, a sentence pair \((f^l, e^l)\) will be formally split into a segmentation \(S^1_{ij}K\) which consists of \(K\) phrase pairs. Each \(s_k = (i_k, b_k, j_k)\) is a triple consisting of the last position \(i_k\) of the \(k\)th target phrase \(e_k\). The start and end position of the \(k\)th source phrase \(f_k\) are \(b_k\) and \(j_k\). Suppose the search state is now extended with a new phrase pair \((f_k, e_k)\): \(\tilde{f}_k := f_{b_k} \ldots f_{j_k}\) and \(\tilde{e}_k := e_{i_k-1} \ldots e_{i_k}\). We have access to the old coverage vector, from which we know if the new phrase’s left neighboring source word \(f_{b_k-1}\) and right neighboring source word \(f_{j_k+1}\) have been translated. We also have the word alignment within the new phrase pair, which is stored during the phrase extraction process. Based on the old coverage vector and alignment, we can repeat the transformation in Figure 1 to calculate the labels for the new phrase. The added model will then check the consistency between the calculated labels and the labels predicted by the reordering model. The number of source words that have inconsistent labels is the penalty and is then added into the log-linear framework as a new feature.

4 Comparative Study

The second part of this paper is comparative study on reordering models. Here we briefly describe those models which will be compared to later.

4.1 Moses lexicalized reordering model

Moses (Koehn et al., 2007) contains a word-based reordering orientation model, which has three types of reordering: (m) monotone order, (s) switch with previous phrase and (d) discontinuous. Figure 2 is an example. The definitions of reordering types are as follows:

- **Monotone** for current phrase, if a word alignment to the bottom left (point A) exists and there is no word alignment point at the bottom right position (point B).
- **Swap** for current phrase, if a word alignment to the bottom right (point B) exists and there is no word alignment point at the bottom left position (point A).
- **Discontinuous** all other cases

Our implementation is same with the default behavior of Moses lexicalized reordering model. We count how often each extracted phrase pair is found with each of the three reordering types. The add-0.5 smoothing is then applied. Finally, the probability is estimated with maximum likelihood principle.

4.2 Maximum entropy reordering model

Figure 3 is an illustration of (Zens and Ney, 2006) . \(j\) is the source word position which is aligned to the last target word of the current phrase. \(j'\) is the last source word position of the current phrase. \(j''\) is the source word position which is aligned to the first target word position of the next phrase. (Zens and Ney, 2006) proposed a maximum entropy classifier to predict the orientation of the next phrase given the current phrase. The orientation class \(c_{j,j''-j'}\) is defined as:

\[
c_{j,j''-j'} = \begin{cases} 
\text{left}, & \text{if } j'' < j \\
\text{right}, & \text{if } j'' > j \text{ and } j'' - j' > 1 \\
\text{monotone}, & \text{if } j'' > j \text{ and } j'' - j' = 1
\end{cases}
\]

(4)

The orientation probability is modeled in a log-linear framework using a set of \(N\) feature functions \(h_n(f^l, e^l, i, j, c_{j,j'',j''}, n = 1, \ldots, N\). The whole model is:

\[
p_N(c_{j,j'',j''} | f^l, e^l, i, j) = \frac{\exp\left(\sum_{n=1}^{N} \lambda_n h_n(f^l, e^l, i, j, c_{j,j'',j''})\right)}{\sum' \exp\left(\sum_{n=1}^{N} \lambda_n h_n(f^l, e^l, i, j, c'')\right)}
\]

(5)

Different features can be used, we use the source and target word features to train the model.
4.3 Bilingual LM

The previous two models belong to “The reordering is a classification problem”. Now we turn to “The reordering is a decoding order problem”. (Mariño et al., 2006) implement a translation model using n-grams. In this way, the translation system can take full advantage of the smoothing and consistency provided by standard back-off n-gram models. Figure 4 is an example. The interpretation is that given the sentence pair \((f_1, e_1)\) and its alignment, the correct translation order is 
\[ e_1, e_2, f_3, e_3, f_1, e_4, f_4, e_5, f_6, f_7, e_7, f_5. \]

Notice the bilingual units have been ordered according to the target side, as the decoder writes the translation in a left-to-right way. Using the example we describe the strategies used for special cases:

- keep the unaligned target word, e.g. \(e_1\)
- remove the unaligned source word, e.g. \(f_2\)
- when one source word aligned to multiple target words, duplicate the source word for each target word, e.g. \(e_4, f_4\), \(e_5, f_4\)
- when multiple source words aligned to one target word, put together the source words for that target word, e.g. \(e_6, f_6, f_7\)

After the operation in Figure 4 was done for all bilingual sentence pairs, we get a decoding sequence. In search, the bilingual LM is applied similar to the standard target side LM. The bilingual sequence of phrase pairs will be extracted using the same strategy in Figure 4. Suppose the search state is now extended with a new phrase pair \((\tilde{f}, \tilde{e})\). \(\tilde{F}\) is the bilingual sequence for the new phrase pair \((\tilde{f}, \tilde{e})\) and \(F^i\) is the \(i^{th}\) unit within \(\tilde{F}\). \(\tilde{F}'\) is the bilingual sequence history for current state. We compute the feature score \(h_{bilm}(\tilde{F}, \tilde{F}')\) of the extended state as follows:

\[
h_{bilm}(\tilde{F}, \tilde{F}') = \lambda \sum_{i=1}^{\lvert \tilde{F} \rvert} \log p(\tilde{F}^i | \tilde{F}', \tilde{F}^{i-1})
\]

\(\lambda\) is the scaling factor for this model. \(\lvert \tilde{F} \rvert\) is the length of the bilingual decoding sequence.

4.4 Source decoding sequence LM

(Feng et al., 2010) present an simpler version of the above bilingual LM where they use only the source side to model the decoding order. The source word decoding sequence in Figure 4 is then \(f_3, f_1, f_2, f_4, f_6, f_7, f_5\). We also build a 9-gram LM based on the source word decoding sequences. The usage of the model is same as bilingual LM.

4.5 Syntactic cohesion model

The previous two models belong to “The reordering is a decoding order problem”. Now we turn to “The reordering can be solved by outside heuristics”. (Cherry, 2008) proposed a syntactic cohesion model. The core idea is that the syntactic structure of the source sentence should be preserved during translation. This structure is represented by a source sentence dependency tree. The algorithm is as follows: given the source sentence and its dependency tree, during the translation process, once a hypothesis is extended, check if the source dependency tree contains a subtree \(T\) such that:
5 Experiments

In this section, we describe the baseline setup, the CRFs training results, the RNN training results and translation experimental results.

5.1 Experimental Setup

Our baseline is a phrase-based decoder, which includes the following models: an \( n \)-gram target-side language model (LM), a phrase translation model and a word-based lexicon model. The latter two models are used for both directions: \( p(f|e) \) and \( p(e|f) \). Additionally we use phrase count features, word and phrase penalty. The reordering model for the baseline system is the distance-based jump model which uses linear distance. This model does not have hard limit. We list the important information regarding the experimental setup below. All those conditions have been kept same in this work.

- lowercased training data from the GALE task (Table 1, UN corpus not included)
- alignment trained with GIZA++
- tuning corpus: NIST06 test corpora: NIST02 03 04 05 and 08
- 5-gram LM (1 694 412 027 running words) trained by SRILM toolkit (Stolcke, 2002) with modified Kneser-Ney smoothing training data: target side of bilingual data.
- BLeU (Papineni et al., 2001) and TER (Snover et al., 2005) reported all scores calculated in lowercase way.
- Wapiti toolkit (Lavergne et al., 2010) used for CRFs; RNN is built by the RNNGLIB toolkit.

<table>
<thead>
<tr>
<th>Sentences</th>
<th>Chinese</th>
<th>English</th>
</tr>
</thead>
<tbody>
<tr>
<td>Running Words</td>
<td>115 172 748</td>
<td>129 820 318</td>
</tr>
<tr>
<td>Vocabulary</td>
<td>1 125 437</td>
<td>739 251</td>
</tr>
</tbody>
</table>

Table 1: translation model and LM training data statistics

Table 1 contains the data statistics used for translation model and LM. For the reordering model, we take two further filtering steps. Firstly, we delete the sentence pairs if the source sentence length is one. When the source sentence has only one word, the translation will be always monotonic and the reordering model does not need to learn this. Secondly, we delete the sentence pairs if the source sentence contains more than three contiguous unaligned words. When this happens, the sentence pair is usually low quality hence not suitable for learning. The main purpose of the two filtering steps is to further lay down the computational burden. The label distribution is depicted in Figure 5. We can see that most words are monotonic. We then divide the corpus to three parts:
5.2 CRFs Training Results

The toolkit Wapiti (Lavergne et al., 2010) is used in this paper. We choose the classical optimization algorithm limited memory BFGS (L-BFGS) (Liu and Nocedal, 1989). For regularization, Wapiti uses both the $\ell^1$ and $\ell^2$ penalty terms, yielding the elastic-net penalty of the form

$$
\rho_1 \cdot \| \theta \|_1 + \frac{\rho^2}{2} \cdot \| \theta \|_2^2
$$

In this work, we use as many features as possible because $\ell^1$ penalty $\rho_1 \cdot \| \theta \|_1$ is able to yield sparse parameter vectors, i.e. using a $\ell^1$ penalty term implicitly performs the feature selection. The computational costs are given here: on a cluster with two AMD Opteron(tm) Processor 6176 (total 24 cores), the training time is about 16 hours, peak memory is around 120G. Several experiments have been done to find the suitable hyperparameter $\rho_1$ and $\rho_2$, we choose the model with lowest error rate on validation corpus for translation experiments. The error rate of the chosen model on test corpus (the test corpus in Table 2) is 25.75% for token error rate and 69.39% for sequence error rate. Table 3 is the feature template we set initially which generates 722999637 features. Some examples are given in Table 4. After training 36902363 features are kept.

5.3 RNN Training Results

We also applied RNN to the task as an alternative approach to CRFs. The here used RNN implementation is RNLLIB which has support for long short term memory (LSTM) (Graves, 2008). We used a one of k encoding for the input word and also for the labels. After testing several configurations over the validation corpus we used a network with 25 hidden units, peak memory is around 120G. Several experiments have been done to find the suitable hyperparameter $\rho_1$ and $\rho_2$, we choose the model with lowest error rate on validation corpus for translation experiments. The error rate of the chosen model on test corpus (the test corpus in Table 2) is 25.75% for token error rate and 69.39% for sequence error rate. Table 3 is the feature template we set initially which generates 722999637 features. Some examples are given in Table 4. After training 36902363 features are kept.

### Table 2: tagging-style model training data statistics

<table>
<thead>
<tr>
<th>Sentences</th>
<th>train</th>
<th>validation</th>
<th>test</th>
</tr>
</thead>
<tbody>
<tr>
<td>Running Words</td>
<td>2973519</td>
<td>400000</td>
<td>400000</td>
</tr>
<tr>
<td>Vocabulary</td>
<td>62263.295</td>
<td>8370.361</td>
<td>8382.086</td>
</tr>
</tbody>
</table>

### Table 4: feature examples. \texttt{x[\text{row},\text{col}]} specifies a token in the input data. \texttt{row} specifies the relative position from the current label and \texttt{col} specifies the absolute position of the column. So for the current label in this table, \texttt{x[-1,2]/x[0,2]} is Lmono-Rmono/UNALIGN and \texttt{x[-1,1]/x[0,1]/x[1,1]} is PN/VC/VV.

<table>
<thead>
<tr>
<th>Words</th>
<th>POS</th>
<th>Label</th>
<th>Feature Templates</th>
</tr>
</thead>
<tbody>
<tr>
<td>基于</td>
<td>P</td>
<td>BEGIN-Rmono</td>
<td>31%</td>
</tr>
<tr>
<td>这</td>
<td>DT</td>
<td>Lmono-Rmono</td>
<td>31%</td>
</tr>
<tr>
<td>看法</td>
<td>N</td>
<td>Lmono-Rmono</td>
<td>31%</td>
</tr>
<tr>
<td>,</td>
<td>PU</td>
<td>Lmono-Rmono</td>
<td>31%</td>
</tr>
<tr>
<td>本</td>
<td>PN</td>
<td>Lmono-Rmono</td>
<td>31%</td>
</tr>
<tr>
<td>是</td>
<td>VC</td>
<td>UNALIGN</td>
<td>31%</td>
</tr>
<tr>
<td>支持</td>
<td>VV</td>
<td>Lmono-Rmono</td>
<td>31%</td>
</tr>
<tr>
<td>修正案</td>
<td>NN</td>
<td>Lmono-Rmono</td>
<td>31%</td>
</tr>
<tr>
<td>的</td>
<td>DEC</td>
<td>UNALIGN</td>
<td>31%</td>
</tr>
<tr>
<td>:</td>
<td>PU</td>
<td>END-Lmono</td>
<td>31%</td>
</tr>
</tbody>
</table>

Table 4: feature examples. \texttt{x[\text{row},\text{col}]} specifies a token in the input data. \texttt{row} specifies the relative position from the current label and \texttt{col} specifies the absolute position of the column. So for the current label in this table, \texttt{x[-1,2]/x[0,2]} is Lmono-Rmono/UNALIGN and \texttt{x[-1,1]/x[0,1]/x[1,1]} is PN/VC/VV.

LSTM 200 nodes in the hidden layer. The RNN has a token error rate of 27.31% and a sentence error rate of 77.00% over the test corpus in Table 2. The RNN is trained on a similar computer as above. RNLLIB utilizes only one thread. The training time is about three and a half days and peak memory consumption is 1G.

5.4 Comparison of CRFs and RNN errors

CRFs performs better than RNN (token error rate 25.75% vs 27.31%). Both error rate values are much higher than what we usually see in part-of-speech tagging task. The main reason is that the “annotated” corpus is converted from word alignment which contains lots of error. However, as we
will show later, the model trained with both CRFs and RNN help to improve the translation quality.

Table 5 and Table 6 demonstrate the confusion matrix of the CRFs and RNN errors over the test corpus. The rows represent the correct tag that the classifier should have predicted and the columns are the actually predicted tags. E.g. the number 687724 in first row and first column of Table 5 tells that there are 687724 correctly labeled Unalign tags. The number 15084 in first row and second column of Table 5 represents that there are 15084 Unalign tags labeled incorrectly to Begin-Rmono. Therefore, numbers on the diagonal from the upper left to the lower right corner represent the amount of correctly classified tags and all other numbers show the amount of false labels. The many zeros show that both classifier rarely make mistake for the label “BEGIN-*” which only occur at the beginning of a sentence. The same is true for the “END-*” labels.

### 5.5 Translation Results

Results are summarized in Table 7. Please read the caption for the meaning of abbreviations. An Index column is added for score reference convenience (B for BLEU; T for TER). For the proposed model, significance testing results on both BLEU and TER are reported (B2 and B3 compared to B1, T2 and T3 compared to T1). We perform bootstrap resampling with bounds estimation as described in (Koehn, 2004). The 95% confidence threshold (denoted by ‡ in the table) is used to draw significance conclusions. We add a column avg. to show the average improvements.

From Table 7 we see that the proposed reordering model using CRFs improves the baseline by 0.98 BLEU and 1.21 TER on average, while the proposed reordering model using RNN improves the baseline by 1.32 BLEU and 1.53 TER on average. For line B2 B3 and T2 T3, most scores are better than their corresponding baseline values with more than 95% confidence. The results show that our proposed idea improves the baseline system and RNN trained model performs better than CRFs trained model, in terms of both automatic measure and significance test. To investigate why RNN has lower performance for the tagging task but achieves better BLEU, we build a 3-gram LM on the source side of the training corpus in Table 2 and perplexity values are listed in Table 8. The perplexity of the test corpus for reordering model comparison is much lower than those NIST corpora for translation experiments. In other words, there exists mismatch of the data for reordering model training and actual MT data. This could explain why CRFs is superior to RNN for labeling problem while RNN is better for MT tasks.

For the comparative study, the best method is the tree-based jump model (JUMPTREE). Our proposed model ranks the second position. The difference is tiny: on average only 0.08 BLEU (B3 and B10) and 0.15 TER (T3 and T10). Even with
We choose CRFs and RNN to accomplish the tagging-style model trained with CRFs or RNN; LRM for maximum entropy reordering model (Zens and Ney, 2006); BILM for bilingual language model (Mariño et al., 2006) and SRCLM for its simpler version source decoding sequence model (Feng et al., 2010); SC for syntactic cohesion model (Cherry, 2008); SRL for semantic cohesion model (Feng et al., 2012); JUMPTREE for our tree-based jump model based on (Wang et al., 2007).

Table 7: Experimental results. CRFs and RNN mean the tagging-style model trained with CRFs or RNN; LRM for maximum entropy reordering model (Koehn et al., 2007); MERO for maximum entropy reordering model (Zens and Ney, 2006); BILM for bilingual language model (Mariño et al., 2006) and SRCLM for its simpler version source decoding sequence model (Feng et al., 2010); SC for syntactic cohesion model (Cherry, 2008); SRL for semantic cohesion model (Feng et al., 2012); JUMPTREE for our tree-based jump model based on (Wang et al., 2007).

Table 8: Perplexity

Table: a strong system (B11 and T11), our model is still able to provide improvements (B12 and T12).

6 Conclusion

In this paper, a novel tagging style reordering model has been proposed. By our method, the reordering problem is converted into a sequence labeling problem so that the whole source sentence is taken into consideration for reordering decision. By adding an unaligned word tag, the unaligned word phenomenon is automatically implanted in the proposed model. The model is utilized as soft constraints in the decoder. In practice, we do not experience decoding memory increase nor speed slow down.

We choose CRFs and RNN to accomplish the sequence labeling task. The CRFs achieves lower error rate on the tagging task but RNN trained model is better for the translation task. Experimental results show that our model is stable and improves the baseline system by 0.98 BLEU and 1.21 TER (trained by CRFs) and 1.32 BLEU and 1.53 TER (trained by RNN). Most of the scores are better than their corresponding baseline values with more than 95% confidence. We also compare our method with several other popular reordering models. Our model ranks the second position which is slightly worse than the tree-based jump model. However, the tree-based jump model relies on manually designed reordering rules which does not exist for many language pairs while our model can be easily adapted to other translation tasks. We also show that the proposed model is able to improve a very strong baseline system.

The main contributions of the paper are: propose the tagging-style reordering model and improve the translation quality; compare two sequence labeling techniques CRFs and RNN; compare our method with seven other reordering models. To our best knowledge, it is the first time that the above two comparisons have been reported.

Acknowledgments

This work was partly realized as part of the Quaero Programme, funded by OSEO, French State agency for innovation, and also partly based upon work supported by the Defense Advanced Research Projects Agency (DARPA) under Contract No. HR0011-12-C-0015. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the Defense Advanced Research Projects Agency (DARPA).
References


