Inducing a Discriminative Parser to Optimize Machine Translation Reordering

Graham Neubig[1,2], **Taro Watanabe**[2], **Shinsuke Mori**[1]
[1]Graduate School of Informatics, Kyoto University
Yoshida Honmachi, Sakyo-ku, Kyoto, Japan
[2]National Institute of Information and Communication Technology
3-5 Hikari-dai, Seika-cho, Soraku-gun, Kyoto, Japan

Abstract

This paper proposes a method for learning a discriminative parser for machine translation reordering using only aligned parallel text. This is done by treating the parser's derivation tree as a latent variable in a model that is trained to maximize reordering accuracy. We demonstrate that efficient large-margin training is possible by showing that two measures of reordering accuracy can be factored over the parse tree. Using this model in the pre-ordering framework results in significant gains in translation accuracy over standard phrase-based SMT and previously proposed unsupervised syntax induction methods.

1 Introduction

Finding the appropriate word ordering in the target language is one of the most difficult problems for statistical machine translation (SMT), particularly for language pairs with widely divergent syntax. As a result, there is a large amount of previous research that handles the problem of reordering through the use of improved reordering models for phrase-based SMT (Koehn et al., 2005), hierarchical phrase-based translation (Chiang, 2007), syntax-based translation (Yamada and Knight, 2001), or pre-ordering (Xia and McCord, 2004).

In particular, systems that use source-language syntax allow for the handling of long-distance reordering without large increases in decoding time. However, these require a good syntactic parser, which is not available for many languages. In recent work, DeNero and Uszkoreit (2011) suggest that unsupervised grammar induction can be used to create source-sentence parse structure for use in translation as a part of a pre-ordering based translation system.

In this work, we present a method for inducing a parser for SMT by training a discriminative model to maximize reordering accuracy while treating the parse tree as a latent variable. As a learning framework, we use online large-margin methods to train the model to directly minimize two measures of reordering accuracy. We propose a variety of features, and demonstrate that learning can succeed when no linguistic information (POS tags or parse structure) is available in the source language, but also show that this linguistic information can be simply incorporated when it is available. Experiments find that the proposed model improves both reordering and translation accuracy, leading to average gains of 1.2 BLEU points on English-Japanese and Japanese-English translation without linguistic analysis tools, or up to 1.5 BLEU points when these tools are incorporated. In addition, we show that our model is able to effectively maximize various measures of reordering accuracy, and that the reordering measure that we choose has a direct effect on translation results.

2 Preordering for SMT

Machine translation is defined as transformation of source sentence $F = f_1 \dots f_J$ to target sentence $E = e_1 \dots e_I$. In this paper, we take

The first author is now affiliated with the Nara Institute of Science and Technology.

Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, pages 843–853, Jeju Island, Korea, 12–14 July 2012.

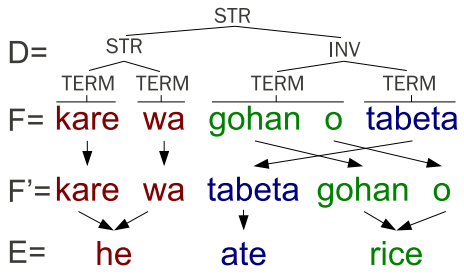

Figure 1: An example with a source sentence F reordered into target order F', and its corresponding target sentence E. D is one of the BTG derivations that can produce this ordering.

the pre-ordering approach to machine translation (Xia and McCord, 2004), which performs translation as a two step process of reordering and translation (Figure 1). Reordering first deterministically transforms F into F', which contains the same words as F but is in the order of E. Translation then transforms F' into E using a method such as phrase-based SMT (Koehn et al., 2003), which can produce accurate translations when only local reordering is required.

This general framework has been widely studied, with the majority of works relying on a syntactic parser being available in the source language. Reordering rules are defined over this parse either through machine learning techniques (Xia and McCord, 2004; Zhang et al., 2007; Li et al., 2007; Genzel, 2010; Dyer and Resnik, 2010; Khalilov and Sima'an, 2011) or linguistically motivated manual rules (Collins et al., 2005; Xu et al., 2009; Carpuat et al., 2010; Isozaki et al., 2010b). However, as building a parser for each source language is a resource-intensive undertaking, there has also been some interest in developing reordering rules without the use of a parser (Rottmann and Vogel, 2007; Tromble and Eisner, 2009; DeNero and Uszkoreit, 2011; Visweswariah et al., 2011), and we will follow this thread of research in this paper.

In particular, two methods deserve mention for being similar to our approach. First, DeNero and Uszkoreit (2011) learn a reordering model through a three-step process of bilingual grammar induction, training a monolingual parser to reproduce the induced trees, and training a reordering model that selects a reordering based on this parse structure. In contrast, our method trains the model in a single step, treating the parse structure as a latent variable in a discriminative reordering model. In addition Tromble and Eisner (2009) and Visweswariah et al. (2011) present models that use binary classification to decide whether each pair of words should be placed in forward or reverse order. In contrast, our method uses traditional context-free-grammar models, which allows for simple parsing and flexible parameterization, including features such as those that utilize the existence of a span in the phrase table. Our work is also unique in that we show that it is possible to directly optimize several measures of reordering accuracy, which proves important for achieving good translations.[1]

3 Training a Reordering Model with Latent Derivations

In this section, we provide a basic overview of the proposed method for learning a reordering model with latent derivations using online discriminative learning.

3.1 Space of Reorderings

The model we present here is based on the bracketing transduction grammar (BTG, Wu (1997)) framework. BTGs represent a binary tree derivation D over the source sentence F as shown in Figure 1. Each non-terminal node can either be a straight (STR) or inverted (INV) production, and terminals (TERM) span a non-empty substring $\boldsymbol{f}$.[2]

The ordering of the sentence is determined by the tree structure and the non-terminal labels STR and INV, and can be built bottom-up. Each subtree represents a source substring $\boldsymbol{f}$ and its reordered counterpart $\boldsymbol{f}'$. For each terminal node, no reordering occurs and $\boldsymbol{f}$ is equal to $\boldsymbol{f}'$.

[1] The semi-supervised method of Katz-Brown et al. (2011) also optimizes reordering accuracy, but requires manually annotated parses as seed data.

[2] In the original BTG framework used in translation, terminals produce a bilingual substring pair $\boldsymbol{f}/\boldsymbol{e}$, but as we are only interested in reordering the source F, we simplify the model by removing the target substring $\boldsymbol{e}$.

For each non-terminal node spanning $\boldsymbol{f}$ with its left child spanning $\boldsymbol{f}_1$ and its right child spanning $\boldsymbol{f}_2$, if the non-terminal symbol is STR, the reordered strings will be concatenated in order as $\boldsymbol{f}' = \boldsymbol{f}'_1\boldsymbol{f}'_2$, and if the non-terminal symbol is INV, the reordered strings will be concatenated in inverted order as $\boldsymbol{f}' = \boldsymbol{f}'_2\boldsymbol{f}'_1$.

We define the space of all reorderings that can be produced by the BTG as $\mathcal{F}'$, and attempt to find the best reordering $\hat{F}'$ within this space.[3]

3.2 Reorderings with Latent Derivations

In order to find the best reordering $\hat{F}'$ given only the information in the source side sentence F, we define a scoring function $S(F'|F)$, and choose the ordering of maximal score:

$$\dot{F}' = \arg\max_{F'} S(F'|F).$$

As our model is based on reorderings licensed by BTG derivations, we also assume that there is an underlying derivation D that produced F'. As we can uniquely determine F' given F and D, we can define a scoring function $S(D|F)$ over derivations, find the derivation of maximal score

$$\dot{D} = \arg\max_{D} S(D|F)$$

and use $\dot{D}$ to transform F into F'.

Furthermore, we assume that the score $S(D|F)$ is the weighted sum of a number of feature functions defined over D and F

$$S(D|F, \boldsymbol{w}) = \sum_i w_i \phi_i(D, F)$$

where ϕ_i is the ith feature function, and w_i is its corresponding weight in weight vector $\boldsymbol{w}$.

Given this model, we must next consider how to learn the weights $\boldsymbol{w}$. As the final goal of our model is to produce good reorderings F', it is natural to attempt to learn weights that will allow us to produce these high-quality reorderings.

[3]BTGs cannot reproduce all possible reorderings, but can handle most reorderings occurring in natural translated text (Haghighi et al., 2009).

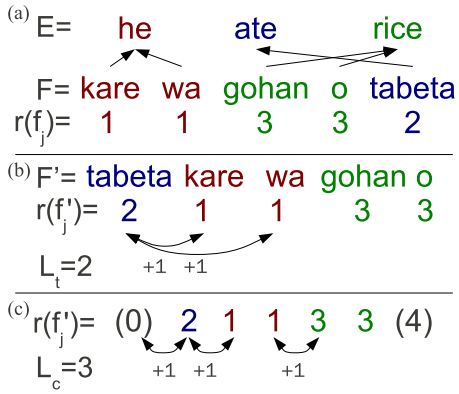

Figure 2: An example of (a) the ranking function $r(f_j)$, (b) loss according to Kendall's τ, (c) loss according to chunk fragmentation.

4 Evaluating Reorderings

Before we explain the learning algorithm, we must know how to distinguish whether the F' produced by the model is good or bad. This section explains how to calculate oracle reorderings, and assign each F' a *loss* and an *accuracy* according to how well it reproduces the oracle.

4.1 Calculating Oracle Orderings

In order to calculate reordering quality, we first define a ranking function $r(f_j|F, A)$, which indicates the relative position of source word f_j in the proper target order (Figure 2 (a)). In order to calculate this ranking function, we define $A = \boldsymbol{a}_1, \ldots, \boldsymbol{a}_J$, where each $\boldsymbol{a}_j$ is a set of the indices of the words in E to which f_j is aligned.[4] Given these alignments, we define an ordering function $\boldsymbol{a}_{j_1} < \boldsymbol{a}_{j_2}$ that indicates that the indices in $\boldsymbol{a}_{j_1}$ come before the indices in $\boldsymbol{a}_{j_2}$. Formally, we define this function as "the first index in $\boldsymbol{a}_{j_1}$ is at most the first index in $\boldsymbol{a}_{j_2}$, similarly for the last index, and either the first or last index in $\boldsymbol{a}_{j_1}$ is less than that of $\boldsymbol{a}_{j_2}$."

Given this ordering, we can sort every alignment $\boldsymbol{a}_j$, and use its relative position in the sentence to assign a rank to its word $r(f_j)$. In

[4]Null alignments require special treatment. To do so, we can place unaligned brackets and quotes directly before and after the spans they surround, and attach all other unaligned words to the word directly to the right for head-initial languages (e.g. English), or left for head-final languages (e.g. Japanese).

the case of ties, where neither $\boldsymbol{a}_{j_1} < \boldsymbol{a}_{j_2}$ nor $\boldsymbol{a}_{j_2} < \boldsymbol{a}_{j_1}$, both f_{j_1} and f_{j_2} are assigned the same rank. We can now define measures of reordering accuracy for F' by how well it arranges the words in order of ascending rank. It should be noted that as we allow ties in rank, there are multiple possible F' where all words are in strictly ascending order, which we will call *oracle* orderings.

4.2 Kendall's τ

The first measure of reordering accuracy that we will consider is Kendall's τ (Kendall, 1938), a measure of pairwise rank correlation which has been proposed for evaluating translation reordering accuracy (Isozaki et al., 2010a; Birch et al., 2010) and pre-ordering accuracy (Talbot et al., 2011). The fundamental idea behind the measure lies in comparisons between each pair of elements f'_{j_1} and f'_{j_2} of the reordered sentence, where $j_1 < j_2$. Because $j_1 < j_2$, f'_{j_1} comes before f'_{j_2} in the reordered sentence, the ranks should be $r(f'_{j_1}) \leq r(f'_{j_2})$ in order to produce the correct ordering.

Based on this criterion, we first define a loss $L_t(F')$ that will be higher for orderings that are further from the oracle. Specifically, we take the sum of all pairwise orderings that do not follow the expected order

$$L_t(F') = \sum_{j_1=1}^{J-1} \sum_{j_2=j_1+1}^{J} \delta(r(f'_{j_1}) > r(f'_{j_2}))$$

where $\delta(\cdot)$ is an indicator function that is 1 when its condition is true, and 0 otherwise. An example of this is given in Figure 2 (b).

To calculate an accuracy measure for ordering F', we first calculate the maximum loss for the sentence, which is equal to the total number of non-equal rank comparisons in the sentence[5]

$$\max_{F'} L_t(F') = \sum_{j_1=1}^{J-1} \sum_{j_2=j_1+1}^{J} \delta(r(f'_{j_1}) \neq r(f'_{j_2})). \quad (1)$$

[5]The traditional formulation of Kendall's τ assumes no ties in rank, and thus the maximum loss can be calculated as $J(J-1)/2$.

Finally, we use this maximum loss to normalize the actual loss to get an accuracy

$$A_t(F') = 1 - \frac{L_t(F')}{\max_{\tilde{F}'} L_t(\tilde{F}')},$$

which will take a value between 0 (when F' has maximal loss), and 1 (when F' matches one of the oracle orderings). In Figure 2 (b), $L_t(F') = 2$ and $\max_{\tilde{F}'} L_t(\tilde{F}') = 8$, so $A_t(F') = 0.75$.

4.3 Chunk Fragmentation

Another measure that has been used in evaluation of translation accuracy (Banerjee and Lavie, 2005) and pre-ordering accuracy (Talbot et al., 2011) is chunk fragmentation. This measure is based on the number of chunks that the sentence needs to be broken into to reproduce the correct ordering, with a motivation that the number of continuous chunks is equal to the number of times the reader will have to jump to a different position in the reordered sentence to read it in the target order. One way to measure the number of continuous chunks is considering whether each word pair f'_j and f'_{j+1} is discontinuous (the rank of f'_{j+1} is not equal to or one greater than f'_j)

$$\text{DISCONT}(f'_j, f'_{j+1}) = \delta(r(f'_j) \neq r(f'_{j+1}) \wedge r(f'_j) + 1 \neq r(f'_{j+1}))$$

and sum over all word pairs in the sentence to create a sentence-based loss

$$L_c(F') = \sum_{j=1}^{J-1} \text{DISCONT}(f'_j, f'_{j+1}) \quad (2)$$

While this is the formulation taken by previous work, we found that this under-penalizes bad reorderings of the first and last words of the sentence, which can contribute to the loss only once, as opposed to other words which can contribute to the loss twice. To account for this, when calculating the chunk fragmentation score, we additionally add two sentence boundary words f_0 and f_{J+1} with ranks $r(f_0) = 0$ and $r(f_{J+1}) = 1 + \max_{f'_j \in F'} r(f'_j)$ and redefine the summation in Equation (2) to consider these words (e.g. Figure 2 (c)).

procedure WEIGHTUPDATE(F, A, $\boldsymbol{w}$)
  $\mathcal{D} \leftarrow \text{parse}(F, \boldsymbol{w})$ ▷ Create parse forest
  $\dot{D} \leftarrow \underset{D \in \mathcal{D}}{\text{argmax}}\ S(D|F, \boldsymbol{w}) + L(D|F, A)$
  ▷ Find the model parse
  $\hat{D} \leftarrow \underset{D \in \mathcal{D}}{\text{argmin}}\ L(D|F, A) - \alpha S(D|F, \boldsymbol{w})$
  ▷ Find the oracle parse
  if $L(\hat{D}|F, A) \neq L(\dot{D}|F, A)$ **then**
    $\boldsymbol{w} \leftarrow \beta(\boldsymbol{w} + \gamma(\phi(\hat{D}, F) - \phi(\dot{D}, F)))$
    ▷ Perform weight update
  end if
end procedure

Figure 3: An online update for sentence F, alignment A, and weight vector $\boldsymbol{w}$. α is a very small constant, and β and γ are defined by the update strategy.

Similarly to Kendall's τ, we can also define an accuracy measure between 0 and 1 using the maximum loss, which will be at most $J + 1$, which corresponds to the total number of comparisons made in calculating the loss[6]

$$A_c(F') = 1 - \frac{L_c(F')}{J+1}.$$

In Figure 2 (c), $L_c(F') = 3$ and $J + 1 = 6$, so $A_c(F') = 0.5$.

5 Learning a BTG Parser for Reordering

Now that we have a definition of loss over reorderings produced by the model, we have a clear learning objective: we would like to find reorderings F' with low loss. The learning algorithm we use to achieve this goal is motivated by discriminative training for machine translation systems (Liang et al., 2006), and extended to use large-margin training in an online framework (Watanabe et al., 2007).

5.1 Learning Algorithm

Learning uses the general framework of large-margin online structured prediction (Crammer et al., 2006), which makes several passes through the data, finding a derivation with high model score (the *model* parse) and a derivation with minimal loss (the *oracle* parse), and updating $\boldsymbol{w}$ if these two parses diverge (Figure 3).

In order to create both of these parses efficiently, we first create a parse forest encoding a large number of derivations $\mathcal{D}_i$ according to the model scores. Next, we find the model parse $\dot{D}_i$, which is the parse in the forest $\mathcal{D}_i$ that maximizes the sum of the model score and the loss $S(D_k|F_k, \boldsymbol{w}) + L(D_k|F_k, A_k)$. It should be noted that here we are considering not only the model score, but also the derivation's loss. This is necessary for loss-driven large-margin training (Crammer et al., 2006), and follows the basic intuition that during training, we would like to make it easier to select negative examples with large loss, causing these examples to be penalized more often and more heavily.

We also find an oracle parse $\hat{D}_i$, which is selected solely to minimize the loss $L(D_k|F_k, A_k)$. One important difference between the model we describe here and traditional parsing models is that the target derivation $\hat{D}_k$ is a latent variable. Because many D_k achieve a particular reordering F', many reorderings F' are able to minimize the loss $L(F'_k|F_k, A_k)$. Thus it is necessary to choose a single oracle derivation to treat as the target out of many equally good reorderings. DeNero and Uszkoreit (2011) resolve this ambiguity with four features with empirically tuned scores before training a monolingual parser and reordering model. In contrast, we follow previous work on discriminative learning with latent variables (Yu and Joachims, 2009), and break ties within the pool of oracle derivations by selecting the derivation with the largest model score. From an implementation point of view, this can be done by finding the derivation that minimizes $L(D_k|F_k, A_k) - \alpha S(D_k|F_k, \boldsymbol{w})$, where α is a constant small enough to ensure that the effect of the loss will always be greater than the effect of the score.

Finally, if the model parse $\dot{D}_k$ has a loss that is greater than that of the oracle parse $\hat{D}_k$, we update the weights to increase the score of the oracle parse and decrease the score of the model parse. Any criterion for weight updates may be used, such as the averaged perceptron (Collins, 2002) and MIRA (Crammer et al., 2006), but

[6] It should be noted that for sentences of length one or sentences with tied ranks, the maximum loss may be less than $J + 1$, but for simplicity we use this approximation.

we opted to use Pegasos (Shalev-Shwartz et al., 2007) as it allows for the introduction of regularization and relatively stable learning.

To perform this full process, given a source sentence F_k, alignment A_k, and model weights $\boldsymbol{w}$ we need to be able to efficiently calculate scores, calculate losses, and create parse forests for derivations D_k, the details of which will be explained in the following sections.

5.2 Scoring Derivation Trees

First, we must consider how to efficiently assign scores $S(D|F, \boldsymbol{w})$ to a derivation or forest during parsing. The most standard and efficient way to do so is to create local features that can be calculated based only on the information included in a single node d in the derivation tree. The score of the whole tree can then be expressed as the sum of the scores from each node:

$$\begin{aligned} S(D|F, \boldsymbol{w}) &= \sum_{d \in D} S(d|F, \boldsymbol{w}) \\ &= \sum_{d \in D} \sum_{i} w_i \phi_i(d, F). \end{aligned}$$

Based on this restriction, we define a number of features that can be used to score the parse tree. To ease explanation, we represent each node in the derivation as $d = \langle s, l, c, c+1, r \rangle$, where s is the node's symbol (STR, INV, or TERM), while l and r are the leftmost and rightmost indices of the span that d covers. c and $c+1$ are the rightmost index of the left child and leftmost index of the right child for non-terminal nodes.

All features are intersected with the node label s, so each feature described below corresponds to three different features (or two for features applicable to only non-terminal nodes).

- $\boldsymbol{\phi}_{lex}$: Identities of words in positions f_l, f_r, f_c, f_{c+1}, f_{l-1}, f_{r+1}, $f_l f_r$, and $f_c f_{c+1}$.
- $\boldsymbol{\phi}_{class}$: Same as $\boldsymbol{\phi}_{lex}$, but with words abstracted to classes. We use the 50 classes automatically generated by Och (1999)'s method that are calculated during alignment in standard SMT systems.
- $\boldsymbol{\phi}_{balance}$: For non-terminals, features indicating whether the length of the left span $(c-l+1)$ is lesser than, equal to, or greater than the length of the right span $(r-c)$.
- $\boldsymbol{\phi}_{table}$: Features, bucketed by length, that indicate whether "$f_l \dots f_r$" appears as a contiguous phrase in the SMT training data, as well as the log frequency of the number of times the phrase appears total and the number of times it appears as a contiguous phrase (DeNero and Uszkoreit, 2011). Phrase length is limited to 8, and phrases of frequency one are removed.
- $\boldsymbol{\phi}_{pos}$: Same as $\boldsymbol{\phi}_{lex}$, but with words abstracted to language-dependent POS tags.
- $\boldsymbol{\phi}_{cfg}$: Features indicating the label of the spans $f_l \dots f_r$, $f_l \dots f_c$, and $f_{c+1} \dots f_r$ in a supervised parse tree, and the intersection of the three labels. When spans do not correspond to a span in the supervised parse tree, we indicate "no span" with the label "X" (Zollmann and Venugopal, 2006).

Most of these features can be calculated from only a parallel corpus, but $\boldsymbol{\phi}_{pos}$ requires a POS tagger and $\boldsymbol{\phi}_{cfg}$ requires a full syntactic parser in the source language. As it is preferable to have a method that is applicable in languages where these tools are not available, we perform experiments both with and without the features that require linguistic analysis tools.

5.3 Finding Losses for Derivation Trees

The above features $\boldsymbol{\phi}$ and their corresponding weights $\boldsymbol{w}$ are all that are needed to calculate scores of derivation trees at test time. However, during training, it is also necessary to find model parses according to the loss-augmented scoring function $S(D|F, \boldsymbol{w}) + L(D|F, A)$ or oracle parses according to the loss $L(D|F, A)$. As noted by Taskar et al. (2003), this is possible if our losses can be factored in the same way as the feature space. In this section, we demonstrate that the loss $L(d|F, A)$ for the evaluation measures we defined in Section 4 can (mostly) be factored over nodes in a fashion similar to features.

5.3.1 Factoring Kendall's τ

For Kendall's τ, in the case of terminal nodes, $L_t(d = \langle \textsc{term}, l, r\rangle | F, A)$ can be calculated by performing the summation in Equation (1). We can further define this sum recursively and use memoization for improved efficiency

$$\begin{aligned} L_t(d|F,A) =& L_t(\langle \textsc{term}, l, r-1\rangle | F, A) \\ &+ \sum_{j=l}^{r-1} \delta(r(f_j) > r(f_r)). \end{aligned} \quad (3)$$

For non-terminal nodes, we first focus on straight non-terminals with parent node $d = \langle \textsc{str}, l, c, c+1, r\rangle$, and left and right child nodes $d_l = \langle s_l, l, lc, lc+1, c\rangle$ and $d_r = \langle s_r, c+1, rc, rc+1, r\rangle$. First, we note that the loss for the subtree rooted at d can be expressed as

$$\begin{aligned} L_t(d|F,A) =& L_t(d_l|F,A) + L_t(d_r|F,A) \\ &+ \sum_{j_1=l}^{c} \sum_{j_2=c+1}^{r} \delta(r(f_{j_1}) > r(f_{j_2})). \end{aligned}$$

In other words, the subtree's total loss can be factored into the loss of its left subtree, the loss of its right subtree, and the additional loss contributed by comparisons between the words spanning both subtrees. In the case of inverted terminals, we must simply reverse the comparison in the final sum to be $\delta(r(f_{j_1}) < r(f_{j_2}))$.

5.3.2 Factoring Chunk Fragmentation

Chunk fragmentation loss can be factored in a similar fashion. First, it is clear that the loss for the terminal nodes can be calculated efficiently in a fashion similar to Equation (3). In order to calculate the loss for non-terminals d, we note that the summation in Equation (2) can be divided into the sum over the internal bi-grams in the left and right subtrees, and the bi-gram spanning the reordered trees

$$\begin{aligned} L_c(d|F,A) =& L_c(d_l|F,A) + L_c(d_r|F,A) \\ &+ \textsc{discont}(f'_c, f'_{c+1}). \end{aligned}$$

However, unlike Kendall's τ, this equation relies not on the ranks of f_c and f_{c+1} in the original sentence, but on the ranks of f'_c and f'_{c+1} in the reordered sentence. In order to keep track of these values, it is necessary to augment each node in the tree to be $d = \langle s, l, c, c+1, r, tl, tr\rangle$ with two additional values tl and tr that indicate the position of the leftmost and rightmost words after reordering. Thus, a straight non-terminal parent d with children $d_l = \langle s_l, l, lc, lc+1, c, tl, tlr\rangle$ and $d_r = \langle s_r, c+1, rc, rc+1, r, trl, tr\rangle$ will have loss as follows

$$\begin{aligned} L_c(d|F,A) =& L_c(d_l|F,A) + L_c(d_r|F,A) \\ &+ \textsc{discont}(f_{tlr}, f_{trl}) \end{aligned}$$

with a similar calculation being possible for inverted non-terminals.

5.4 Parsing Derivation Trees

Finally, we must be able to create a parse forest from which we select model and oracle parses. As all feature functions factor over single nodes, it is possible to find the parse tree with the highest score in $O(J^3)$ time using the CKY algorithm. However, when keeping track of target positions for calculation of chunk fragmentation loss, there are a total of $O(J^5)$ nodes, an unreasonable burden in terms of time and memory. To overcome this problem, we note that this setting is nearly identical to translation using synchronous CFGs with an integrated bigram LM, and thus we can employ cube-pruning to reduce our search space (Chiang, 2007).

6 Experiments

Our experiments test the reordering and translation accuracy of translation systems using the proposed method. As reordering metrics, we use Kendall's τ and chunk fragmentation (Talbot et al., 2011) comparing the system F' and oracle F' calculated with manually created alignments. As translation metrics, we use BLEU (Papineni et al., 2002), as well as RIBES (Isozaki et al., 2010a), which is similar to Kendall's τ, but evaluated on the target sentence E instead of the reordered sentence F'. All scores are the average of three training runs to control for randomness in training (Clark et al., 2011).

For translation, we use Moses (Koehn et al., 2007) with lexicalized reordering (Koehn et al., 2005) in all experiments. We test three types

	en-ja				ja-en			
	Chunk	τ	BLEU	RIBES	Chunk	τ	BLEU	RIBES
ORIG	61.22	73.46	21.87	68.25	66.42	72.99	18.34	65.36
3-STEP	63.51	72.55	21.45	67.66	67.17	73.01	17.78	64.42
3-STEP+ϕ_{pos}	64.28	72.11	21.45	67.44	67.56	74.21	18.18	64.65
3-STEP+ϕ_{cfg}	65.76	75.32	21.67	68.47	67.23	74.06	18.18	64.93
LADER	73.19	78.44	**23.11**	69.86	**75.14**	**79.14**	**19.54**	**66.93**
LADER+ϕ_{pos}	73.97	79.24	**23.32**	69.78	**75.49**	**78.79**	**19.89**	**67.24**
LADER+ϕ_{cfg}	**75.06**	**80.53**	**23.36**	**70.89**	**75.14**	77.80	19.35	66.12

Table 2: Reordering (chunk, τ) and translation (BLEU, RIBES) results for each system. Bold numbers indicate no significant difference from the best system (bootstrap resampling with $p > 0.05$) (Koehn, 2004).

	sent.	word (ja)	word (en)
RM-train	602	14.5k	14.3k
RM-test	555	11.2k	10.4k
TM/LM	329k	6.08M	5.91M
Tune	1166	26.8k	24.3k
Test	1160	28.5k	26.7k

Table 1: The number of sentences and words for training and testing the reordering model (RM), translation model (TM), and language model (LM).

of pre-ordering: original order with $F' \leftarrow F$ (ORIG), pre-orderings learned using the 3-step process of DeNero and Uszkoreit (2011) (3-STEP), and the proposed model with latent derivations (LADER).[7] Except when stated otherwise, LADER was trained to minimize chunk fragmentation loss with a cube pruning stack pop limit of 50, and the regularization constant of 10^{-3} (chosen through cross-validation).

We test our systems on Japanese-English and English-Japanese translation using data from the Kyoto Free Translation Task (Neubig, 2011). We use the training set for training translation and language models, the development set for weight tuning, and the test set for testing (Table 1). We use the designated development and test sets of manually created alignments as training data for the reordering models, removing sentences of more than 60 words.

As default features for LADER and the monolingual parsing and reordering models in 3-STEP, we use all the features described in Section 5.2 except $\boldsymbol{\phi}_{pos}$ and $\boldsymbol{\phi}_{cfg}$. In addition, we test systems with $\boldsymbol{\phi}_{pos}$ and $\boldsymbol{\phi}_{cfg}$ added. For English, we use the Stanford parser (Klein and Manning, 2003) for both POS tagging and CFG parsing. For Japanese, we use the KyTea tagger (Neubig et al., 2011) for POS tagging,[8] and the EDA word-based dependency parser (Flannery et al., 2011) with simple manual head-rules to convert a dependency parse to a CFG parse.

6.1 Effect of Pre-ordering

Table 2 shows reordering and translation results for ORIG, 3-STEP, and LADER. It can be seen that the proposed LADER outperforms the baselines in both reordering and translation.[9] There are a number of reasons why LADER outperforms 3-STEP. First, the pipeline of 3-STEP suffers from error propogation, with errors in monolingual parsing and reordering resulting in low overall accuracy.[10] Second, as Section 5.1 describes, LADER breaks ties between oracle parses based on model score, allowing easy-to-reproduce model parses to be chosen during training. In fact, LADER generally found trees that followed from syntactic constituency, while 3-STEP more often used terminal nodes

[7] Available open-source: `http://phontron.com/lader`

[8] In addition, following the example of Sudoh et al. (2011a)'s reordering rules, we lexicalize all particles.

[9] It should be noted that our results for 3-STEP are significantly worse than those of DeNero and Uszkoreit (2011). Likely reasons include a 20x difference in training data size, the fact that we are using naturally translated text as opposed to text translated specifically to create word alignments, or differences in implementation.

[10] When using oracle parses, chunk accuracy was up to 81%, showing that parsing errors are highly detrimental.

	en-ja				ja-en			
	Chunk	τ	BLEU	RIBES	Chunk	τ	BLEU	RIBES
L_c	**73.19**	78.44	**23.11**	69.86	**75.14**	**79.14**	**19.54**	**66.93**
L_t	70.37	79.57	22.57	69.47	72.51	78.93	18.52	66.26
$L_c + L_t$	72.55	**80.58**	**22.89**	**70.34**	74.44	**79.82**	19.21	**66.48**

Table 3: Results for systems trained to optimize chunk fragmentation (L_c) or Kendall's τ (L_t).

that spanned constituent boundaries (as long as the phrase frequency was high). Finally, as Section 6.2 shows in detail, the ability of LADER to maximize reordering accuracy directly allows for improved reordering and translation results.

It can also be seen that incorporating POS tags or parse trees improves accuracy of both LADER and 3-STEP, particularly for English-Japanese, where syntax has proven useful for pre-ordering, and less so for Japanese-English, where syntactic pre-ordering has been less successful (Sudoh et al., 2011b).

We also tested Moses's implementation of hierarchical phrase-based SMT (Chiang, 2007), which achieved BLEU scores of 23.21 and 19.30 for English-Japanese and Japanese-English respectively, approximately matching LADER in accuracy, but with a significant decrease in decoding speed. Further, when pre-ordering with LADER and hierarchical phrase-based SMT were combined, BLEU scores rose to 23.29 and 19.69, indicating that the two techniques can be combined for further accuracy improvements.

6.2 Effect of Training Loss

Table 3 shows results when one of three losses is optimized during training: chunk fragmentation (L_c), Kendall's τ (L_t), or the linear interpolation of the two with weights chosen so that both losses contribute equally ($L_t + L_c$). In general, training successfully maximizes the criterion it is trained on, and $L_t + L_c$ achieves good results on both measures. We also find that L_c and $L_c + L_t$ achieve the best translation results, which is in concert with Talbot et al. (2011), who find chunk fragmentation is better correlated with translation accuracy than Kendall's τ. This is an important result, as methods such as that of Tromble and Eisner (2009) optimize pairwise word comparisons equivalent to L_t, which may not be optimal for translation.

	en-ja BLEU/RIBES		ja-en BLEU/RIBES	
ORIG	21.87	68.25	18.34	65.36
MAN-602	**23.11**	**69.86**	**19.54**	**66.93**
AUTO-602	22.39	69.19	18.58	66.07
AUTO-10K	22.53	**69.68**	18.79	**66.89**

Table 4: Results based on data size, and whether manual or automatic alignments are used in training.

6.3 Effect of Automatic Alignments

Table 4 shows the difference between using manual and automatic alignments in the training of LADER. LADER is able to improve over the ORIG baseline in all cases, but when equal numbers of manual and automatic alignments are used, the reorderer trained on manual alignments is significantly better. However, as the number of automatic alignments is increased, accuracy improves, approaching that of the system trained on a smaller number of manual alignments.

7 Conclusion

We presented a method for learning a discriminative parser to maximize reordering accuracy for machine translation. Future work includes application to other language pairs, development of more sophisticated features, investigation of probabilistic approaches to inference, and incorporation of the learned trees directly in tree-to-string translation.

Acknowledgments

We thank Isao Goto, Tetsuo Kiso, and anonymous reviewers for their helpful comments, and Daniel Flannery for helping to run his parser.

References

Satanjeev Banerjee and Alon Lavie. 2005. METEOR: An automatic metric for MT evaluation with improved correlation with human judgments. In *Proc. ACL Workshop*.

Alexandra Birch, Miles Osborne, and Phil Blunsom. 2010. Metrics for MT evaluation: evaluating reordering. *Machine Translation*, 24(1):15–26.

Marine Carpuat, Yuval Marton, and Nizar Habash. 2010. Improving arabic-to-english statistical machine translation by reordering post-verbal subjects for alignment. In *Proc. ACL*.

David Chiang. 2007. Hierarchical phrase-based translation. *Computational Linguistics*, 33(2).

Jonathan H. Clark, Chris Dyer, Alon Lavie, and Noah A. Smith. 2011. Better hypothesis testing for statistical machine translation: Controlling for optimizer instability. In *Proc. ACL*, pages 176–181.

Michael Collins, Philipp Koehn, and Ivona Kucerova. 2005. Clause restructuring for statistical machine translation. In *Proc. ACL*.

Michael Collins. 2002. Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms. In *Proc. EMNLP*, pages 1–8.

Koby Crammer, Ofer Dekel, Joseph Keshet, Shai Shalev-Shwartz, and Yoram Singer. 2006. Online passive-aggressive algorithms. *Journal of Machine Learning Research*, 7:551–585.

John DeNero and Jakob Uszkoreit. 2011. Inducing sentence structure from parallel corpora for reordering. In *Proc. EMNLP*.

Chris Dyer and Philip Resnik. 2010. Context-free reordering, finite-state translation. In *Proc. HLT-NAACL*.

Daniel Flannery, Yusuke Miyao, Graham Neubig, and Shinsuke Mori. 2011. Training dependency parsers from partially annotated corpora. In *Proc. IJCNLP*, pages 776–784, Chiang Mai, Thailand, November.

Dmitriy Genzel. 2010. Automatically learning source-side reordering rules for large scale machine translation. In *Proc. COLING*.

Aria Haghighi, John Blitzer, John DeNero, and Dan Klein. 2009. Better word alignments with supervised ITG models. In *Proc. ACL*.

Hideki Isozaki, Tsutomu Hirao, Kevin Duh, Katsuhito Sudoh, and Hajime Tsukada. 2010a. Automatic evaluation of translation quality for distant language pairs. In *Proc. EMNLP*, pages 944–952.

Hideki Isozaki, Katsuhito Sudoh, Hajime Tsukada, and Kevin Duh. 2010b. Head finalization: A simple reordering rule for sov languages. In *Proc. WMT and MetricsMATR*.

Jason Katz-Brown, Slav Petrov, Ryan McDonald, Franz Och, David Talbot, Hiroshi Ichikawa, Masakazu Seno, and Hideto Kazawa. 2011. Training a parser for machine translation reordering. In *Proc. EMNLP*, pages 183–192.

Maurice G. Kendall. 1938. A new measure of rank correlation. *Biometrika*, 30(1/2):81–93.

Maxim Khalilov and Khalil Sima'an. 2011. Context-sensitive syntactic source-reordering by statistical transduction. In *Proc. IJCNLP*.

Dan Klein and Christopher D. Manning. 2003. Accurate unlexicalized parsing. In *Proc. ACL*, pages 423–430.

Phillip Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proc. HLT*, pages 48–54.

Philipp Koehn, Amittai Axelrod, Alexandra Birch Mayne, Chris Callison-Burch, Miles Osborne, and David Talbot. 2005. Edinburgh system description for the 2005 IWSLT speech translation evaluation. In *Proc. IWSLT*.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *Proc. ACL*, pages 177–180.

Philipp Koehn. 2004. Statistical significance tests for machine translation evaluation. In *Proc. EMNLP*.

Chi-Ho Li, Minghui Li, Dongdong Zhang, Mu Li, Ming Zhou, and Yi Guan. 2007. A probabilistic approach to syntax-based reordering for statistical machine translation. In *Proc. ACL*.

Percy Liang, Alexandre Bouchard-Côté, Dan Klein, and Ben Taskar. 2006. An end-to-end discriminative approach to machine translation. In *Proc. ACL*, pages 761–768.

Graham Neubig, Yosuke Nakata, and Shinsuke Mori. 2011. Pointwise prediction for robust, adaptable Japanese morphological analysis. In *Proc. ACL*, pages 529–533, Portland, USA, June.

Graham Neubig. 2011. The Kyoto free translation task. http://www.phontron.com/kftt.

Franz Josef Och. 1999. An efficient method for determining bilingual word classes. In *Proc. EACL*.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. In *Proc. COLING*, pages 311–318.

Kay Rottmann and Stephan Vogel. 2007. Word reordering in statistical machine translation with a pos-based distortion model. In *Proc. of TMI-2007*.

Shai Shalev-Shwartz, Yoram Singer, and Nathan Srebro. 2007. Pegasos: Primal estimated subgradient solver for SVM. In *Proc. ICML*, pages 807–814.

Katsuhito Sudoh, Kevin Duh, Hajime Tsukada, Masaaki Nagata, Xianchao Wu, Takuya Matsuzaki, and Jun'ichi Tsujii. 2011a. NTT-UT statistical machine translation in NTCIR-9 PatentMT. In *Proc. NTCIR*.

Katsuhito Sudoh, Xianchao Wu, Kevin Duh, Hajime Tsukada, and Masaaki Nagata. 2011b. Post-ordering in statistical machine translation. In *Proc. MT Summit*.

David Talbot, Hideto Kazawa, Hiroshi Ichikawa, Jason Katz-Brown, Masakazu Seno, and Franz Och. 2011. A lightweight evaluation framework for machine translation reordering. In *Proc. WMT*.

Ben Taskar, Carlos Guestrin, and Daphne Koller. 2003. Max-margin Markov networks. *Proc. NIPS*, 16.

Roy Tromble and Jason Eisner. 2009. Learning linear ordering problems for better translation. In *Proc. EMNLP*.

Karthik Visweswariah, Rajakrishnan Rajkumar, Ankur Gandhe, Ananthakrishnan Ramanathan, and Jiri Navratil. 2011. A word reordering model for improved machine translation. In *Proc. EMNLP*.

Taro Watanabe, Jun Suzuki, Hajime Tsukada, and Hideki Isozaki. 2007. Online large-margin training for statistical machine translation. In *Proc. EMNLP*, pages 764–773.

Dekai Wu. 1997. Stochastic inversion transduction grammars and bilingual parsing of parallel corpora. *Computational Linguistics*, 23(3).

Fei Xia and Michael McCord. 2004. Improving a statistical MT system with automatically learned rewrite patterns. In *Proc. COLING*.

Peng Xu, Jaeho Kang, Michael Ringgaard, and Franz Och. 2009. Using a dependency parser to improve smt for subject-object-verb languages. In *Proc. NAACL*.

Kenji Yamada and Kevin Knight. 2001. A syntax-based statistical translation model. In *Proc. ACL*.

Chun-Nam John Yu and Thorsten Joachims. 2009. Learning structural SVMs with latent variables. In *Proc. ICML*, pages 1169–1176.

Yuqi Zhang, Richard Zens, and Hermann Ney. 2007. Chunk-level reordering of source language sentences with automatically learned rules for statistical machine translation. In *Proc. SSST*.

Andreas Zollmann and Ashish Venugopal. 2006. Syntax augmented machine translation via chart parsing. In *Proc. WMT*, pages 138–141.