Delimiting Morphosyntactic Search Space with Source-Side Reordering Models

Joachim Daiber and **Khalil Sima'an**
Institute for Logic, Language and Computation
University of Amsterdam
Science Park 107, 1098 XG Amsterdam, The Netherlands
{j.daiber,k.simaan}@uva.nl

Abstract

Source-side reordering has recently seen a surge in popularity in machine translation research, often providing enormous reductions in translation time and showing good empirical results in translation quality. For many language pairs, however—especially for translation into morphologically rich languages—the assumptions of these models may be too crude. But while such language pairs call for more complex models, these could increase the search space to an extent that would diminish their benefits. In this paper, we examine the question whether purely syntax-oriented adaptation models (i.e., models only considering word order) can be used as a means to delimit the search space for more complex morphosyntactic models. We propose a model based on a popular preordering algorithm (Lerner and Petrov, 2013). This novel preordering model is able to produce both *n*-best word order predictions as well as distributions over possible word order choices in the form of a lattice and is therefore a good fit for use by richer models taking into account aspects of both syntax and morphology. We show that the integration of non-local language model features can be beneficial for the model's preordering quality and evaluate the space of potential word order choices the model produces.

1 Introduction

A significant amount of research in machine translation has recently focused on methods for effectively restricting the often prohibitively large search space of statistical machine translation systems. One popular method providing a crude but theoretically motivated restriction of this space is preordering (also pre-reordering or source-side reordering). In preordering, the source sentence is rearranged to reflect the assumed word order in the target language. This provides an effective method for handling word and phrase movements caused by long-range dependencies, which usually enlarge the search space significantly. After preordering, decoding can be performed in fully monotone or close to monotone fashion, making the method applicable to a wide range of translation systems, including ngram-based translation (Marino et al., 2006) and recent approaches to neural machine translation (Bahdanau et al., 2015, inter alia). While systems using this approach have in the past not always been able to show improvements in translation quality over systems using more exhaustive search algorithms or specialized reordering models, preordering provides several benefits: Apart from facilitating the integration of additional information sources such as paraphrases, preordering approaches provide significant improvements in runtime performance. Jehl et al. (2014), for example, report an 80-fold speed improvement using their preordering system compared to a standard system producing translations of the same quality.

Preordering systems can be compared along several dimensions. The main distinctions are whether the reordering rules are specified manually (Collins et al., 2005) or automatically learnt from data (Lerner and Petrov, 2013; Khalilov and Sima'an, 2012). Furthermore, approaches differ in the types of syntactic structures they assume. Systems may use either source or target syntax (Lerner and Petrov, 2013; Khalilov and Sima'an, 2012), both source and target syntax or no syntax at all (e.g. DeNero and Uszkoreit (2011)). In this paper, we focus on approaches using only source-side syntax. Dependency grammar offers a flexible and light-weight syntactic framework that can cover a large number of languages and

Proceedings of the 1st Deep Machine Translation Workshop (DMTW 2015), pages 29–38,
Praha, Czech Republic, 3–4 September 2015.

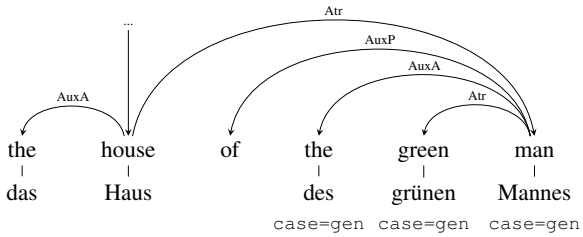

Figure 1: Translation of an English prepositional phrase as a genitive noun phrase in German.

provides suitable syntactic representations for reordering. Hence, we follow Lerner and Petrov (2013), in using dependency trees for the representation of source syntax. After reviewing related work in Section 2, we propose a model and general framework for producing the space of potential word order choices in Section 3. Since the possible reorderings in source-syntax approaches to preordering are often restricted by the source trees, the annotation conventions of the training treebank and hence the form of the predicted dependency trees play a significant role for the preordering system. We will briefly describe the treebank format and other details of the experimental setup in Section 4.1. Section 4.2 and 4.3 present results of the experimental evaluation and a discussion of these results. We conclude in Section 5.

2 Related work

Various approaches to preordering have been explored in the literature. A brief overview of the work establishing the background for the presented method will be given in this section.

2.1 Gold experiments

To investigate the upper bounds of preordering in terms of quality and integration with translation systems, several researchers have performed studies with gold reorderings. Khalilov and Sima'an (2012), as well as Herrmann et al. (2013) compare various systems and provide oracle scores for syntax-based preordering models. These studies show that *perfect gold* reorderings estimated via automatic alignments enable translation systems enormous jumps in translation quality and further provide improvements in the size of the downstream translation models. Additionally, it was found that properties of the source syntax representation, such as how deeply phrase structure trees are nested, can significantly hamper the quality of these approaches.

2.2 Preordering with source syntax

Jehl et al. (2014) learn order decisions for sibling nodes of the source-side parse tree and explore the space of possible permutations using a depth-first branch-and-bound search. In later work, this model is further improved by replacing the logistic regression classifier with a feed-forward neural network (de Gispert et al., 2015). This modification shows both improved empirical results and eliminates the need for feature engineering. Similarly, Lerner and Petrov (2013) learn classifiers to permute the tree nodes of a dependency tree. The main difference here is that the permutation of up to 6 tree nodes is predicted directly instead of predicting the orientation of individual node pairs. Figure 1 shows an example dependency tree that can serve as input to such systems.

2.3 Preordering without source syntax

Tromble and Eisner (2009) apply machine learning techniques to learn ITG-like orientations (straight or inverted order) for each pair of input words in the sentence. The best reordering is then determined using a standard $O(n^3)$ chart parsing algorithm. Generally, systems not relying on syntactic information fill the full spectrum from simple to advanced approaches. A simple approach is the application of multiple MT systems (Costa-jussà and Fonollosa, 2006): one MT system learns the preordering (i.e. the translation of the source sentence to its preordered form, $\mathbf{s} \rightarrow \mathbf{s}'$) and the second MT system learns to translate the preordered form into the target sentence ($\mathbf{s}' \rightarrow \mathbf{t}$). More advanced approaches include the automatic induction of parse trees from aligned data (DeNero and Uszkoreit, 2011).

3 Generating the space of potential word order choices

3.1 Going beyond first-best predictions

Our work is related to the work of Lerner and Petrov (2013), in which feature-rich discriminative classifiers are trained to directly predict the target-side word order based on source-side dependency trees. This is done by traversing the dependency tree in a top-down fashion and predicting the target order for each tree family (a family consists of a syntactic head and its children). To address sparsity issues, two models are introduced. For each subtree, the 1-step model directly predicts the target order of the child nodes. Unlike other preordering models, which often restrict the space of possible permutations, e.g. by the permutations permissible under the ITG constraint (Wu, 1997), the space of possible permutations for each subtree is restricted to the k permutations most commonly observed in the data. The blow-up in permutation space with growing numbers of children is addressed by a second model, the 2-step model. This model decreases the number of nodes involved in any single word order decision. A binary classifier (pivot classifier, in analogy to quicksort) first predicts whether a child node should occur to the left or to the right of the head of the subtree. The order of the set of nodes to the left and to the right of the head is then directly predicted as in the 1-step model. In total, the 2-step approach requires one pivot classifier, 5 classifiers for the children on the left and 5 classifiers for the children on the right.

The cascade-of-classifiers approach used by this method (i.e. first predict the pivot, then predict the left and right orders, then recurse) exhibits the problematic characteristic that classification errors occurring near the top of the tree will propagate disproportionately to later decisions. The goal of the present work is to enable the preordering model to pass its decisions to a more complex morphosyntactic model. Hence, this error propagation issue may become problematic. In order to address this problem, we extract *n*-best word order predictions from the classifier decisions. A distribution over the *n*-best preordered sentences can then be passed to the subsequent model or directly to a machine translation decoder either as a list of options or in the form of a lattice. Similar to the practice of *n*-best list extraction in MT decoders such as Moses, the preordering problem likewise allows the extraction of *n*-best preordering options either with or without additional integration of non-local features such as a language model.

General model We define a model over the possible orders of the tokens in the source sentence. Given a source sentence $\mathbf{s}$ and a corresponding dependency tree τ, π is the permutation of source tokens and π_h is a local permutation of a single tree family under head h. The score of a source word order $\mathbf{s}'$ is:

$$P(\mathbf{s}' \mid \mathbf{s}, \tau) = \prod_{h \in \tau} P_T(\pi_h \mid \mathbf{s}, h, \tau) \tag{1}$$

$$P_T(\pi \mid \mathbf{s}, h, \tau) = P(\psi \mid \mathbf{s}, h, \tau) P_L(\pi_L \mid \mathbf{s}, h, \tau) P_R(\pi_R \mid \mathbf{s}, h, \tau)$$

For each dependency tree family, the generative story of this model is as follows: First, decide on the positions of the child nodes relative to the head, i.e. $P(\psi \mid \mathbf{s}, h, \tau)$. Then, decide the order of the nodes on the left, $P_L(\pi_L \mid \mathbf{s}, h, \tau)$, and on the right, $P_R(\pi_R \mid \mathbf{s}, h, \tau)$.

Preordering algorithm Based on this model, we introduce the following preordering algorithm. For each source dependency tree family with head h, we extract the best k_T local word order predictions using the function PREORDERFAMILY in Algorithm 1. $\Psi(cs)$ is the set of possible choices when distributing nodes using the pivot classifier. Given a set of child nodes cs, $\Pi(cs)$ is the set of their possible permutations. The best permutations for the left and right side are extracted by the following methods:

$$\hat{\pi}_L \leftarrow \underset{\pi_L \in \Pi(cs_L)}{\arg \text{bestk}} P_L(\pi_L \mid \mathbf{s}, h, \tau) \tag{2}$$

$$\hat{\pi}_R \leftarrow \underset{\pi_R \in \Pi(cs_R)}{\arg \text{bestk}} P_R(\pi_R \mid \mathbf{s}, h, \tau) \tag{3}$$

Since this model is implemented using multi-class classifiers, finding the best k_O permutations for the nodes to the left and right of the head, i.e. Equation 2 and 3, only require one multi-class classification. Following Lerner and Petrov (2013), we restrict the set of allowed permutations $\Pi(cs)$ to the 20 most common permutations observed in the training data. Given a pivot decision $\hat{\psi}$ (which children go left and

which go right of the head?), LEFT($\hat{\psi}$) returns the children to the left and RIGHT($\hat{\psi}$) returns the children to the right of the head. The function PERMUTATION($\hat{\psi}, \hat{\pi}_L, \hat{\pi}_R$) returns the word order permutation resulting from the pivot decision, the left children order and the right children order.

Algorithm 1 *n*-best preordering of a source tree family

procedure PREORDERFAMILY(h, τ)
  $cs \leftarrow$ CHILDREN(h, τ)
  $topk \leftarrow$ PRIORITYQUEUE()

  for $\hat{\psi} \leftarrow \underset{\psi \in \Psi(cs)}{\arg\,\text{bestk}}\ P(\psi \mid \mathbf{s}, h, \tau)$ **do** ▷ Pivot decisions
    $cs_L \leftarrow$ LEFT($\hat{\psi}$)
    $cs_R \leftarrow$ RIGHT($\hat{\psi}$)
    for $\hat{\pi}_L \leftarrow \underset{\pi_L \in \Pi(cs_L)}{\arg\,\text{bestk}}\ P_L(\pi_L \mid \mathbf{s}, h, \tau)$ **do** ▷ Left order decisions
      for $\hat{\pi}_R \leftarrow \underset{\pi_R \in \Pi(cs_R)}{\arg\,\text{bestk}}\ P_R(\pi_R \mid \mathbf{s}, h, \tau)$ **do** ▷ Right order decisions
        $p \leftarrow$ PERMUTATION($\hat{\psi}, \hat{\pi}_L, \hat{\pi}_R$)
        TOPK.PUSH($P(\hat{\psi} \mid \mathbf{s}, h, \tau) \times P_L(\hat{\pi}_L \mid \mathbf{s}, h, \tau) \times P_R(\hat{\pi}_R \mid \mathbf{s}, h, \tau), p$)
  return TOPK.TAKE(k_T)

For n children, there are $S(n, 2)$ possible pivot decisions, where $S(n, k)$ is the Stirling number of the second kind. Since this number grows exponentially with n, it would be extremely expensive, if not infeasible, to consider all possible pivot decisions. Hence, similar to the extraction of $\hat{\pi}_L$ and $\hat{\pi}_R$, the extraction of the possible choices for the pivot decision, i.e. $\hat{\psi}$, is implemented as k-best Viterbi extraction from a conditional random field classifier: $\hat{\psi} \leftarrow \underset{\psi \in \Psi(cs)}{\arg\,\text{bestk}}\ P(\psi \mid \mathbf{s}, h, \tau)$.

This approximation means that only the best k_P pivot decisions are considered. Hence, for each of the maximally k_P possible ways to distribute the child nodes when taking the pivot decision, two classifications have to be performed: one for the nodes on the left and one for the nodes on the right. The extraction of *n*-best word order predictions therefore requires $2 \times k_P$ classifications for each source-side tree family. With the best k_T local permutations for each source tree family, we can then extract *n*-best permutations for the whole tree. If all order decisions in this model are local to their tree family, extracting the best permutations for the whole sentence is straight-forward. In the next section, we will discuss how this assumption changes with the introduction of non-local features.

3.2 Integration of non-local features

While the basic model introduced by Lerner and Petrov (2013) shows promising empirical performance, it also makes fairly strong independence assumptions. The generative process assumes that the local order decisions occur only within individual tree families defined by the dependency tree. Hence, a local word order decision at any point in the dependency tree is fully independent from any other decision in the tree. For languages such as German, this independence assumption can be problematic because the position of a constituent in the sentence bracket influences the internal word order (Müller, 2015). For example, certain positions allow for scrambling, i.e. more or less free movement of some constituents within a specific area of the sentence. Previous work on preordering (Khalilov and Sima'an, 2012) has shown that the integration of even a weak trigram language model estimated over the gold word order predictions $\mathbf{s}'$ can improve preordering performance. Since we use projective dependency trees, which are internally converted to a flat phrase structure representation, the model can be expressed in the form of a weighted context-free grammar in which labels encode the order of the constituents. One method to weaken the independence assumptions of this grammar is the direct integration of a language model (LM). This idea is reminiscent of the integration of the finite state language model with the synchronous context-free grammar used in hierarchical phrase-based machine translation (Chiang, 2007).

Hence, instead of searching for $\hat{\mathbf{s}}' = \arg\max_{\mathbf{s}'} P(\mathbf{s}' \mid \mathbf{s}, \tau)$, the search will now include the ngram language model, such that: $\hat{\mathbf{s}}' = \arg\max_{\mathbf{s}'} P(\mathbf{s}' \mid \mathbf{s}, \tau) P_{LM}(\mathbf{s}')$. This integration can be performed in three ways: the simplest form of integration, which is fast but allows for significant search errors, is to generate an *n*-best list of word order predictions using the −LM preordering model (i.e., without the LM or other non-local features) and re-score this list using the language model. On the other end of the spectrum, the language model can be integrated by performing a full intersection between the preordering CFG and the finite state automaton that defines the language model (Bar-Hillel et al., 1961). While this would allow for exact search, this method is found to be too slow in practice. A compromise between these two extremes is cube pruning (Chiang, 2007), in which the inner LM cost as well as the left and right LM states are stored on each node, so that it is possible to perform bottom-up dynamic programming to efficiently determine the total LM cost by combining the intermediate node costs. Keeping the properties required for performing cube pruning, we use the more general log-linear model formulation (Och and Ney, 2002) by defining the search for the best word order prediction $\hat{\mathbf{s}}'$ as follows:

$$\hat{\mathbf{s}}' = \arg\max_{\mathbf{s}'} P(\mathbf{s}' \mid \mathbf{s}, \tau)^{\lambda_{RM}} P_{LM}(\mathbf{s})^{\lambda_{LM}} ... = \arg\max_{\mathbf{s}'} \prod_i \phi_i(\mathbf{s}')^{\lambda_i} = \arg\max_{\mathbf{s}'} \sum_i \lambda_i \log \phi_i(\mathbf{s}')$$

On every source tree node, cube pruning is performed with a beam size of $k_{+\text{LM}}$ word order predictions. The best $k_{-\text{LM}}$ preordering labels are considered for expansion. Additionally, we prune all preordering labels for which the language model cost is higher than the language model cost of the original source tree order (i.e., performing no reordering). To make individual configurations comparable, we follow Chiang (2007) in adding a heuristic cost that approximates the cost of the first $m-1$ words: $\log P_{\text{LM}}(e_1 ... e_l)$ where $l = \min\{m-1, |e|\}$ for an m-gram language model. In our case, e is the vector of preordered source-side words at a specific tree node. We add the heuristic cost of all relevant feature functions ϕ_i for the set of language model feature functions Φ_{LM} as $\sum_{i \in \Phi_{\text{LM}}} \lambda_i \log \phi_i(e_1 ... e_l)$.

Feature functions

The log-linear model formulation makes the addition of arbitrary local and non-local features possible; i.e., any suitable feature function can be added to this model. We use the following initial features:

Lexicalized preordering model The most important feature is the lexicalized preordering model $P(\mathbf{s}' \mid \mathbf{s}, \tau)$ introduced in Section 3.1. It is lexicalized since it makes decisions based on the source words while other models might make predictions based on non-lexical information (e.g., POS tags).

Language models To weaken the strong independence assumptions of this model, we add a generic ngram language model over the gold word order predictions $\mathbf{s}'$, a language model over part-of-speech tags and a class-based language model.

Unlexicalized preordering model As the lexicalized preordering model might run into sparsity issues, we add as a further feature function a weaker model $P_W(\pi \mid h, cs)$, where cs is the set of children represented by their dependency label and by whether they have children, and h is the head represented by its POS tag. The model is estimated via maximum likelihood estimation from the oracle word order choices restricted by the source-side dependency trees (*oracle tree reorderings*). These tree-restricted oracle word order choices differ from the free oracle word order choices in that words are not allowed to move out of the constituents of the dependency tree. For example, in the English sentence "the house of the green man" in Figure 1, the word "green" would always be on the same side of "house" as "man" since as a dependent of "man", it will always move with "man" in relation to its grandparent "house".

3.3 Applicability of the model

While we focused on one particular *n*-best preordering method in Section 3.1, the general model introduced in Section 3.2 is applicable to any preordering model over source trees for which *n*-best candidates can be extracted. For example, the pairwise neural network-based method by de Gispert et al. (2015) can be used either by extracting *n*-best decisions directly from the graph or, more efficiently, by applying the CKY algorithm on the space of permutations permissible under ITG (Tromble and Eisner, 2009).

4 Experiments

We perform various experiments to evaluate these ideas. Before providing experimental results and evaluation, we will describe selected details of the preordering system and the experimental setup. Further, we highlight assumptions and decisions that were necessary for training the system.

4.1 Implementation and experimental setup

Source-side syntax For preordering to work reliably, the dependency representation should fulfill certain requirements: *Flatter* trees increase the space of covered permutations while the information in the left-out segmentations may be recoverable by the preordering model. Additionally, whenever reasonable, content-bearing elements should be treated as the head.[1] We use a customized version of the treebank collection and transformation tool HamleDT (Zeman et al., 2012) for this purpose.

Model training For training the model, we mostly follow the process from Lerner and Petrov (2013). Training instances are extracted from the automatically aligned training data based on a small set of manually defined rules. To ensure high quality training data, only subtrees that are fully connected by high confidence alignments are considered. The preordering classifiers are trained on the intersection of high-confidence word alignments and the first-best output of the TurboParser dependency parser (Martins et al., 2009). The alignments are created using the Berkeley aligner[2] with the hard intersection setting. This setting ensures that only high confidence alignment links are produced. While this will lead to a reduction in the number of alignment links, it creates more reliable training data for the preordering model. The dependency parser is trained to produce pseudo-projective dependency trees (Nivre and Nilsson, 2005).[3] Appropriate values for $k_{+\text{LM}}$ and $k_{-\text{LM}}$ are determined using grid search. We found that beam sizes above $k_{+\text{LM}} = 15$ and $k_{-\text{LM}} = 5$ did not improve first-best preordering quality.

Model tuning The set of weights λ for the combination of the preordering model and the language models are optimized for a selected target metric on heldout data. The straight-forward choice for this metric is Kendall τ, which indicates the similarity of the word order of both sides. The Kendall τ distance $d_\tau(\pi, \sigma)$ between two permutations π and σ is defined as (Birch et al., 2010):

$$d_\tau(\pi, \sigma) = 1 - \frac{\sum_{i=1}^{n} \sum_{j=1}^{n} z_{ij}}{Z}$$

$$\text{where } z_{ij} = \begin{cases} 1 & \text{if } \pi(i) < \pi(j) \text{ and } \sigma(i) > \sigma(j) \\ 0 & \text{otherwise} \end{cases} \quad \text{and } Z = \frac{(n^2 - n)}{2}$$

The metric indicates the ratio of pairwise order differences between two permutations. An alternative to this ordering measure is the simulation of a full machine translation system, as first proposed by Tromble and Eisner (2009). To ensure that the changes in word order do not affect this mock translation system and to limit its complexity, the system is limited to phrases of length 1.

Tuning is performed using the *tuning as ranking* (PRO) framework (Hopkins and May, 2011). At tuning time, $k_{-\text{LM}}$ and $k_{+\text{LM}}$ are set to 15 and 100 respectively. *PRO* requires the unweighted values of all feature functions; hence, during tuning only, we remember the unweighted feature values on each node and sum over intermediate values to arrive at the overall scores. Training instances for ranking are sampled from the best 100 word order predictions for each sentence in the tuning set. We perform 6 iterations and interpolate the weights of each iteration with the weights from the previous iteration by the recommended factor of $\Psi = 0.1$.

Translation setup To evaluate the model in a full translation setup, we follow the standard approach to preordering. Given the source side $\mathbf{s}$ and the target side $\mathbf{t}$ of the parallel training corpus, we first perform

[1] For example, auxiliary verbs should modify the finite verb and prepositions depend on the head of the noun phrase.

[2] `https://code.google.com/p/berkeleyaligner/`

[3] Projectivization was performed using MaltParser version 1.8; `http://www.maltparser.org/`.

Model	Kendall τ	BLEU ($\hat{\mathbf{s}}' \rightarrow \mathbf{s}'$)
First-best $-$LM	92.16	68.1
First-best $+$LM (cube pruned)	92.27	68.7
Best out of *n*-best $+$LM (cube pruned, $n = 5$)	93.33	–
Best out of *n*-best $+$LM (cube pruned, $n = 10$)	93.72	–

Table 1: LM integration tested on first-best prediction (*en–de*, scores from predicted to gold-ordered *en*).

word alignment using MGIZA++ (Gao and Vogel, 2008). We perform 6 iterations of IBM model 1 training followed by 6 iterations of HMM word alignment and 3 iterations each of IBM model 3 and 4.

After initial training, the preordering model is applied to $\mathbf{s}$, obtaining the preordered corpus $\hat{\mathbf{s}}'$. Since the word order differences between $\hat{\mathbf{s}}'$ and $\mathbf{t}$ should be less acute, less computationally expensive word alignment tools are sufficient to re-align the corpus. We align $\hat{\mathbf{s}}'$ and $\mathbf{t}$ using `fast_align`,[4] an efficient re-parameterization of IBM model 2 (Dyer et al., 2013). Improvements in word order can lead to improvements in alignments and hence the training and word alignment process can be performed repeatedly. Lerner and Petrov (2013) report no significant improvements after the initial re-alignment. Accordingly, we do not iterate the training process either. The underlying translation system is Moses (Koehn et al., 2007) using the standard feature setup and using only the distortion-based reordering model. Tuning is performed using MERT (Och, 2003). The system is trained on the full parallel sections of the Europarl corpus (Koehn, 2005) and tuned and tested on the WMT 2009 and WMT 2010 newstest sets respectively. The language model is a 5-gram ngram model trained on the target side of Europarl and the news commentary corpus.[5]

4.2 Testing the effectiveness of non-local features

While our preliminary results showed that the integration of a language model might be helpful, we now consider this question in more detail. To test whether the language model features are beneficial to the reordering model, we compare two versions of the same system: *first-best* $-$LM is the reordering system without a language model and *first-best* $+$LM is the same system with the language model integrated via cube pruning. Results are presented in Table 1. While Kendall τ gives an impression of the overall word order quality, the BLEU metric gives an indication of the quality of reorderings within the more restricted space of the length of the ngrams used in the metric. The results show that the integration of the language model helps the system improve the quality of the reorderings. We expected the language model to provide benefits mostly on the borders between tree nodes. The BLEU score indicates an improvement in the ordering of short word sequences, which hints at the presence of this benefit.

In the *n*-best $+$LM setup, we produce the top n word order predictions and select the prediction that provides the most Kendall τ improvement. These results hint at the potential improvement contained in the best n predictions of the model. Next, we turn to examining the quality of the space of word order predictions in more detail by applying them in a machine translation task.

4.3 Evaluating the quality of the word order predictions

Our goal in this work has been to use a syntax-oriented preordering model to delimit the search space for a subsequent, more complex model. Hence, in order to examine the model presented in Section 3, we determine the quality of the *n*-best predictions the model produces. We perform the following experiment for the language pair English–German: Using the preordering system, we produce the 10 best word order predictions for each sentence in the test set. We then translate each sentence arranged according to each of the word order predictions using a standard phrase-based machine translation system trained on the corpus produced by the first-best preordering system. After the translation is performed, one translation is selected based on the best sentence-level BLEU score. Table 2 shows results for this setup and for a baseline system without preordering. Both systems use a distortion limit of 7 and use only the standard

[4] `http://github.com/clab/fast_align`

[5] `http://statmt.org/wmt13/translation-task.html`

	Distortion limit	BLEU	METEOR	TER
Baseline	7	15.20	35.43	66.62
Best out of k ($k = 10$)		17.26*	37.97*	62.64*

* Result is statistically significant against baseline at $p < 0.05$.

Table 2: Estimation of the quality of the *k* best word order predictions.

distance-based reordering model. Statistical significance tests are performed using bootstrap resampling (Koehn, 2004) and statistically significant results ($p < 0.05$) are marked with an asterisk. These results show that significant improvements in translation quality measured in terms of BLEU, METEOR and TER are possible based on the space of word order choices provided by our model.

4.4 Discussion

Having introduced our preordering method and having evaluated the influence of non-local features, we are now interested in two basic aspects of the output space provided by this system:

The first aspect is the quality of the space of the space delimited by the preordering system. Since we plan to pass the output space to a richer model, it has to be ensured that a sufficient number of good candidates are contained in this space. This question is answered by the translation experiments performed in Section 4.3, which indicate that even within the first 10 word order predictions, there are enough good instances to enable a significant improvement in translation quality. Since the evaluation of our translation experiments is performed using only automatic evaluation metrics, it is difficult to pinpoint the exact source of these potential improvements. In order to examine the gains in more detail and to determine how much the fluency of the output increased, we therefore intend to perform manual evaluation in future work. The second question is whether the size of the space of potential word order choices is manageable for subsequent models. Since the previous experiments showed that even with only 10 word order predictions, a significant improvement can be observed, it is clear that this very small space can be used by a subsequent model. In addition to this, the output in the form of a lattice allows for using more options and efficient processing using dynamic programming algorithms. Since the model from Section 3.1 works on local tree families in a chart, it may be able to work with a parse forest instead of a tree, possibly alleviating parse errors on the source. We plan to explore this direction in future work.

5 Conclusion

Source-side reordering provides a significant potential for improvements in translation quality and translation performance in machine translation, which was shown in previous studies and is further supported by the method's recent surge in popularity. It is therefore an attractive model to extend to morphosyntax beyond pure word order patterns. Most of the benefits of source-side reordering are due to enabling the modeling of much larger reordering spaces in a more reliable manner than it would be possible within the underlying machine translation system. For languages such as German or Arabic, however, word order and morphology are interconnected and should not be treated in isolation. As a first step towards broader morphosyntactic processing beyond word order only, this paper has explored how a preordering model can be utilized to produce a space of sensible word order predictions. We have presented a novel preordering model for this purpose and have evaluated its outputs with translation experiments using a common system setup. The experiments also show that non-local language model features integrated via cube pruning improve the preordering quality for the language pair English–German. Further, our translation experiments show that this preordering system, when optimized for producing *n*-best predictions, provides an output space that is valuable for further processing both in its compactness and in the potential improvement in translation quality it enables.

Acknowledgements We thank the three anonymous reviewers for their constructive comments and suggestions. The first author is supported by the EXPERT (EXPloiting Empirical appRoaches to Translation) Initial Training Network (ITN) of the European Union's Seventh Framework Programme.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. *Proceedings of the International Conference on Learning Representations*.

Yehoshua Bar-Hillel, M. Perles, and E. Shamir. 1961. On formal properties of simple phrase structure grammars. *Zeitschrift für Phonetik, Sprachwissenschaft und Kommunikationsforschung*, 14:143–172. Reprinted in Y. Bar-Hillel. (1964). *Language and Information: Selected Essays on their Theory and Application*, Addison-Wesley 1964, 116–150.

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

David Chiang. 2007. Hierarchical phrase-based translation. *Computational Linguistics*, 33(2):201–228.

Michael Collins, Philipp Koehn, and Ivona Kucerova. 2005. Clause restructuring for statistical machine translation. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 531–540, Ann Arbor, Michigan, June. Association for Computational Linguistics.

Marta R. Costa-jussà and José A. R. Fonollosa. 2006. Statistical machine reordering. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, pages 70–76, Sydney, Australia, July. Association for Computational Linguistics.

Adrià de Gispert, Gonzalo Iglesias, and William Byrne. 2015. Fast and accurate preordering for SMT using neural networks. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics - Human Language Technologies (NAACL HLT 2015)*, June.

John DeNero and Jakob Uszkoreit. 2011. Inducing sentence structure from parallel corpora for reordering. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 193–203, Edinburgh, Scotland, UK., July. Association for Computational Linguistics.

Chris Dyer, Victor Chahuneau, and Noah A. Smith. 2013. A simple, fast, and effective reparameterization of IBM model 2. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 644–648, Atlanta, Georgia, June. Association for Computational Linguistics.

Qin Gao and Stephan Vogel. 2008. Parallel implementations of word alignment tool. In *Software Engineering, Testing, and Quality Assurance for Natural Language Processing*, pages 49–57. Association for Computational Linguistics.

Teresa Herrmann, Jochen Weiner, Jan Niehues, and Alex Waibel. 2013. Analyzing the potential of source sentence reordering in statistical machine translation. In *Proceedings of the International Workshop on Spoken Language Translation (IWSLT 2013)*.

Mark Hopkins and Jonathan May. 2011. Tuning as ranking. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 1352–1362, Edinburgh, Scotland, UK., July. Association for Computational Linguistics.

Laura Jehl, Adrià de Gispert, Mark Hopkins, and Bill Byrne. 2014. Source-side preordering for translation using logistic regression and depth-first branch-and-bound search. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 239–248, Gothenburg, Sweden, April. Association for Computational Linguistics.

Maxim Khalilov and Khalil Sima'an. 2012. Statistical translation after source reordering: Oracles, context-aware models, and empirical analysis. *Natural Language Engineering*, 18:491–519, 10.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, et al. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th annual meeting of the ACL on interactive poster and demonstration sessions*, pages 177–180. Association for Computational Linguistics.

Philipp Koehn. 2004. Statistical significance tests for machine translation evaluation. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*, pages 388–395. Association for Computational Linguistics.

Philipp Koehn. 2005. Europarl: A parallel corpus for statistical machine translation. In *MT summit*, volume 5, pages 79–86.

Uri Lerner and Slav Petrov. 2013. Source-side classifier preordering for machine translation. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 513–523, Seattle, Washington, USA, October. Association for Computational Linguistics.

José B Marino, Rafael E Banchs, Josep M Crego, Adria de Gispert, Patrik Lambert, José AR Fonollosa, and Marta R Costa-Jussà. 2006. N-gram-based machine translation. *Computational Linguistics*, 32(4):527–549.

Andre Martins, Noah Smith, and Eric Xing. 2009. Concise integer linear programming formulations for dependency parsing. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 342–350, Suntec, Singapore, August. Association for Computational Linguistics.

Stefan Müller. 2015. *Grammatical Theory: From Transformational Grammar to Constraint-Based Approaches*. Number 1 in Lecture Notes in Language Sciences. Language Science Press, Berlin. Open Review Version.

Joakim Nivre and Jens Nilsson. 2005. Pseudo-projective dependency parsing. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 99–106, Ann Arbor, Michigan, June. Association for Computational Linguistics.

Franz Josef Och and Hermann Ney. 2002. Discriminative training and maximum entropy models for statistical machine translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ACL '02, pages 295–302, Stroudsburg, PA, USA. Association for Computational Linguistics.

Franz Josef Och. 2003. Minimum error rate training in statistical machine translation. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics-Volume 1*, pages 160–167. Association for Computational Linguistics.

Roy Tromble and Jason Eisner. 2009. Learning linear ordering problems for better translation. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 1007–1016, Singapore, August. Association for Computational Linguistics.

Dekai Wu. 1997. Stochastic inversion transduction grammars and bilingual parsing of parallel corpora. *Computational Linguistics*, 23(3):377–403.

Daniel Zeman, David Mareček, Martin Popel, Loganathan Ramasamy, Jan Štěpánek, Zdeněk Žabokrtský, and Jan Hajič. 2012. HamleDT: To parse or not to parse? In Nicoletta Calzolari (Conference Chair), Khalid Choukri, Thierry Declerck, Mehmet Uğur Doğan, Bente Maegaard, Joseph Mariani, Jan Odijk, and Stelios Piperidis, editors, *Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC'12)*, Istanbul, Turkey, may. European Language Resources Association (ELRA).