Reordering Grammar Induction

Miloš Stanojević
ILLC
University of Amsterdam
m.stanojevic@uva.nl

Khalil Sima'an
ILLC
University of Amsterdam
k.simaan@uva.nl

Abstract

We present a novel approach for unsupervised induction of a Reordering Grammar using a modified form of permutation trees (Zhang and Gildea, 2007), which we apply to preordering in phrase-based machine translation. Unlike previous approaches, we induce in one step both the hierarchical structure and the transduction function over it from word-aligned parallel corpora. Furthermore, our model (1) handles non-ITG reordering patterns (up to 5-ary branching), (2) is learned from all derivations by treating not only labeling but also bracketing as latent variable, (3) is entirely unlexicalized at the level of reordering rules, and (4) requires no linguistic annotation.

Our model is evaluated both for accuracy in predicting target order, and for its impact on translation quality. We report significant performance gains over phrase reordering, and over two known preordering baselines for English-Japanese.

1 Introduction

Preordering (Collins et al., 2005) aims at permuting the words of a source sentence $\mathbf{s}$ into a new order $\acute{\mathbf{s}}$, hopefully close to a plausible target word order. Preordering is often used to bridge long distance reorderings (e.g., in Japanese- or German-English), before applying phrase-based models (Koehn et al., 2007). Preordering is often broken down into two steps: finding a suitable tree structure, and then finding a transduction function over it. A common approach is to use monolingual syntactic trees and focus on finding a transduction function of the sibling subtrees under the nodes (Lerner and Petrov, 2013; Xia and Mccord, 2004). The (direct correspondence) assumption underlying this approach is that permuting the siblings of nodes in a source syntactic tree can produce a plausible target order. An alternative approach creates reordering rules manually and then learns the right structure for applying these rules (Katz-Brown et al., 2011). Others attempt learning the transduction structure and the transduction function in two separate, consecutive steps (DeNero and Uszkoreit, 2011). Here we address the challenge of learning both the trees and the transduction functions jointly, in one fell swoop, from word-aligned parallel corpora.

Learning both trees and transductions jointly raises two questions. How to obtain suitable trees for the source sentence and how to learn a distribution over random variables specifically aimed at reordering in a hierarchical model? In this work we solve both challenges by using the factorizations of permutations into Permutation Trees (PETs) (Zhang and Gildea, 2007). As we explain next, PETs can be crucial for exposing the hierarchical reordering patterns found in word-alignments.

We obtain permutations in the training data by segmenting every word-aligned source-target pair into *minimal phrase pairs*; the resulting alignment between minimal phrases is written as a permutation (1:1 and onto) on the source side. Every permutation can be factorized into a *forest* of PETs (over the source sentences) which we use as a *latent treebank* for training a Probabilistic Context-Free Grammar (PCFG) tailor made for preordering as we explain next.

Figure 1 shows two alternative PETs for the same permutation over minimal phrases. The nodes have labels (like $P3142$) which stand for local permutations (called prime permutation) over the child nodes; for example, the root label $P3142$ stands for prime permutation $\langle 3, 1, 4, 2\rangle$, which says that the first child of the root becomes 3^{rd} on the target side, the second becomes 1^{st}, the third becomes 4^{th} and the fourth becomes 2^{nd}. The prime permutations are non-factorizable permutations like $\langle 1,2 \rangle$, $\langle 2,1 \rangle$ and $\langle 2,4,1,3 \rangle$.

We think PETs are suitable for learning preordering for two reasons. Firstly, PETs specify *exactly* the phrase pairs defined by the permutation. Secondly, every permutation is factorizable into prime permutations only (Albert and Atkinson, 2005). Therefore, PETs expose *maximal* sharing between different permutations in terms of both phrases and their reordering. We expect this to be advantageous for learning hierarchical reordering.

For learning preordering, we first extract an *initial* PCFG from the latent treebank of PETs over the source sentences only. We initialize the nonterminal set of this PCFG to the *prime permutations* decorating the PET nodes. Subsequently we split these coarse labels in the same way as latent variable splitting is learned for treebank parsing (Matsuzaki et al., 2005; Prescher, 2005; Petrov et al., 2006; Saluja et al., 2014). Unlike treebank parsing, however, our training treebank is latent because it consists of a whole forest of PETs per training instance (s).

Learning the splits on a latent treebank of PETs results in a *Reordering PCFG* which we use to parse input source sentences into split-decorated trees, i.e., the labels are the splits of prime permutations. After parsing s, we map the splits back on their initial prime permutations, and then retrieve a reordered version $\acute{\mathrm{s}}$ of s. In this sense, our latent splits are *dedicated to reordering*.

We face two technical difficulties alien to work on latent PCFGs in treebank parsing. Firstly, as mentioned above, permutations may factorize into more than one PET (a forest) leading to a latent training treebank.[1] And secondly, after we parse a source string s, we are interested in $\acute{\mathrm{s}}$, the permuted version of s, not in the best derivation/PET. Exact computation is a known NP-Complete problem (Sima'an, 2002). We solve this by a new Minimum-Bayes Risk decoding approach using Kendall reordering score as loss function, which is an efficient measure over permutations (Birch and Osborne, 2011; Isozaki et al., 2010a).

In summary, this paper contributes:

- A novel latent hierarchical source reordering model working over all derivations of PETs
- A label splitting approach based on PCFGs over minimal phrases as terminals, learned from an ambiguous treebank, where the label splits start out from prime permutations.
- A fast Minimum Bayes Risk decoding over Kendall τ reordering score for selecting $\acute{\mathrm{s}}$.

We report results for extensive experiments on English-Japanese showing that our Reordering PCFG gives substantial improvements when used as preordering for phrase-based models, outperforming two existing baselines for this task.

2 PETs and the Hidden Treebank

We aim at learning a PCFG which we will use for parsing source sentences s into synchronous trees, from which we can obtain a reordered source version $\acute{\mathrm{s}}$. Since PCFGs are non-synchronous grammars, we will use the nonterminal labels to encode reordering transductions, i.e., this PCFG is implicitly an SCFG. We can do this because s and $\acute{\mathrm{s}}$ are over the same alphabet.

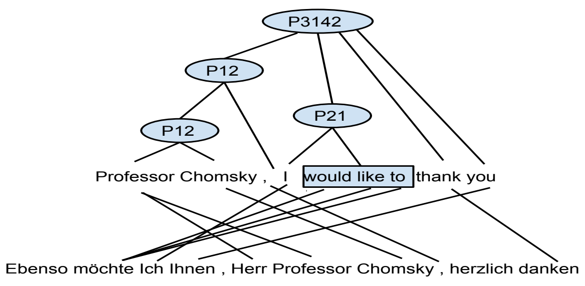
Here, we have access only to a word-aligned parallel corpus, not a treebank. The following steps summarize our approach for acquiring a latent treebank and how it is used for learning a Reordering PCFG:

1. Obtain a permutation over minimal phrases from every word-alignment.
2. Obtain a latent treebank of PETs by factorizing the permutations.
3. Extract a PCFG from the PETs with initial nonterminals taken from the PETs.
4. Learn to split the initial nonterminals and estimate rule probabilities.

These steps are detailed in the next section, but we will start out with an intuitive exposition of PETs, the latent treebank and the Reordering Grammar.

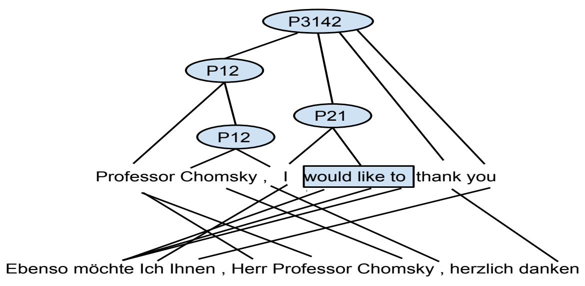
Figure 1 shows examples of how PETs look like – see (Zhang and Gildea, 2007) for algorithmic details. Here we label the nodes with nonterminals which stand for *prime* permutations from the operators on the PETs. For example, nonterminals $P12$, $P21$ and $P3142$ correspond respectively to reordering transducers $\langle 1,2 \rangle$, $\langle 2,1 \rangle$ and $\langle 3,1,4,2 \rangle$. A prime permutation on a source node μ is a transduction dictating how the children of μ are reordered at the target side, e.g., $P21$ inverts the child order. We must stress that any similarity with ITG (Wu, 1997) is restricted to the fact that the straight and inverted operators of ITG are the binary case of prime permutations

[1]All PETs for the same permutation share the same set of prime permutations but differ only in bracketing structure (Zhang and Gildea, 2007).



P3142
P12
P12
P21
Professor Chomsky , I would like to thank you
Ebenso möchte Ich Ihnen , Herr Professor Chomsky , herzlich danken

(a) Canonical PET



P3142
P12
P12
P21
Professor Chomsky , I would like to thank you
Ebenso möchte Ich Ihnen , Herr Professor Chomsky , herzlich danken

(b) Alternative PET

Figure 1: Possible Permutation Trees (PETs) for one sentence pair

in PETs ($P12$ and $P21$). ITGs recognize only the binarizable permutations, which is a major restriction when used on the data: there are many non-binarizable permutations in actual data (Wellington et al., 2006). In contrast, our PETs are obtained by factorizing permutations obtained from the data, i.e., they exactly fit the range of prime permutations in the parallel corpus. In practice we limit them to maximum arity 5.

We can extract PCFG rules from the PETs, e.g., $P21 \rightarrow P12\ P2413$. However, these rules are decorated with too coarse labels. A similar problem was encountered in non-lexicalized monolingual parsing, and one solution was to lexicalize the productions (Collins, 2003) using *head words*. But linguistic heads do not make sense for PETs, so we opt for the alternative approach (Matsuzaki et al., 2005), which splits the nonterminals and softly percolates the splits through the trees gradually fitting them to the training data. Splitting has a shadow side, however, because it leads to combinatorial explosion in grammar size.

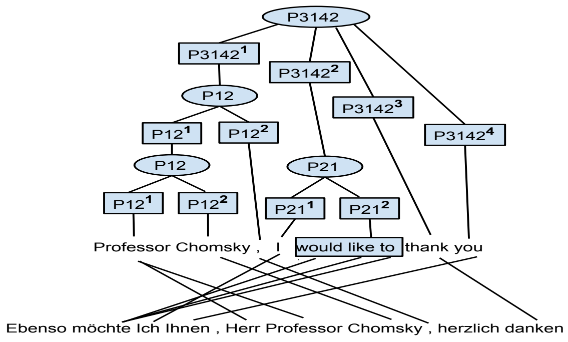
Suppose for example node $P21$ could split into $P21_1$ and $P21_2$ and similarly $P2413$ splits into $P2413_1$ and 2413_2. This means that rule $P21 \rightarrow P12\ P2413$ will form eight new rules:

$$\begin{array}{ll}
P21_1 \rightarrow P12_1\ P2413_1 & P21_1 \rightarrow P12_1\ P2413_2 \\
P21_1 \rightarrow P12_2\ P2413_1 & P21_1 \rightarrow P12_2\ P2413_2 \\
P21_2 \rightarrow P12_1\ P2413_1 & P21_2 \rightarrow P12_1\ P2413_2 \\
P21_2 \rightarrow P12_2\ P2413_1 & P21_2 \rightarrow P12_2\ P2413_2
\end{array}$$

Should we want to split each nonterminal into 30 subcategories, then an *n-ary* rule will split into 30^{n+1} new rules, which is prohibitively large. Here we use the "unary trick" as in Figure 2. The superscript on the nonterminals denotes the child position from left to right. For example $P21_1^2$ means that this node is a second child, and the mother nonterminal label is $P21_1$. For the running example rule, this gives the following rules:

$$\begin{array}{ll}
P21_1 \rightarrow P21_1^1\ P21_1^2 & P21_2 \rightarrow P21_2^1\ P21_2^2 \\
P21_1^1 \rightarrow P12_1 & P21_1^2 \rightarrow P2413_1 \\
P21_1^1 \rightarrow P12_2 & P21_1^2 \rightarrow P2413_2 \\
P21_2^1 \rightarrow P12_1 & P21_2^2 \rightarrow P2413_1 \\
P21_2^1 \rightarrow P12_2 & P21_2^2 \rightarrow P2413_2
\end{array}$$

The unary trick leads to substantial reduction in grammar size, e.g., for arity 5 rules and 30 splits we could have had $30^6 = 729000000$ split-rules, but with the unary trick we only have $30{+}30^2{*}5 = 4530$ split rules. The unary trick was used in early lexicalized parsing work (Carroll and Rooth, 1998).[2] This split PCFG constitutes a *latent PCFG* because the splits cannot be read of a treebank. It must be learned from the latent treebank of PETs, as described next.



P3142
P3142¹
P3142²
P3142³
P3142⁴
P12
P12¹
P12²
P12
P12¹
P12²
P21
P21¹
P21²
Professor Chomsky , I would like to thank you
Ebenso möchte Ich Ihnen , Herr Professor Chomsky , herzlich danken

Figure 2: Permutation Tree with unary trick

3 Details of Latent Reordering PCFG

Obtaining permutations Given a source sentence $\mathbf{s}$ and its alignment $\mathbf{a}$ to a target sentence

[2]After applying the unary trick, we add a constraint on splitting: all nonterminals on an n-ary branching rule must be split simultaneously.

$\mathbf{t}$ in the training corpus, we segment $\langle \mathbf{s}, \mathbf{a}, \mathbf{t} \rangle$ into a sequence of *minimal phrases* $\mathbf{s}_m$ (maximal sequence) such that the reordering between these minimal phrases constitutes a permutation π_m. We do not extract non-contiguous or non-minimal phrases because reordering them often involves complicated transductions which could hamper the performance of our learning algorithm.[3]

Unaligned words Next we describe the use of the factorization of permutations into PET forests for training a PCFG model. But first we need to extend the PETs to allow for unaligned words. An unaligned word is joined with a neighboring phrase to the left or the right, depending on the source language properties (e.g., whether the language is head-initial or -final (Chomsky, 1970)). Our experiments use English as source language (head-initial), so the unaligned words are joined to phrases to their right. This modifies a PET by adding a new binary branching node μ (dominating the unaligned word and the phrase it is joined to) which is labeled with a dedicated nonterminal: $P01$ if the unaligned word joins to the right and $P10$ if it joins to the left.

3.1 Probability model

We decompose the permutation π_m into a forest of permutation trees $PEF(\pi_m)$ in $O(n^3)$, following algorithms in (Zhang et al., 2008; Zhang and Gildea, 2007) with trivial modifications. Each PET $\mathbf{\Delta} \in PEF(\pi_m)$ is a different bracketing (differing in binary branching structure only). We consider the bracketing hidden in the latent treebank, and apply unsupervised learning to induce a distribution over possible bracketings. Our probability model starts from the joint probability of a sequence of minimal phrases $\mathbf{s}_m$ and a permutation π_m over it. This demands summing over all PETs $\mathbf{\Delta}$ in the forest $PEF(\pi_m)$, and for every PET also over all its label splits, which are given by the grammar derivations $\mathbf{d}$:

$$P(\mathbf{s}_m, \pi_m) = \sum_{\mathbf{\Delta} \in PEF(\pi_m)} \sum_{\mathbf{d} \in \mathbf{\Delta}} P(\mathbf{d}, \mathbf{s}_m) \qquad (1)$$

The probability of a derivation $\mathbf{d}$ is a product of probabilities of all the rules $\mathbf{r}$ that build it:

$$P(\mathbf{s}_m, \pi_m) = \sum_{\mathbf{\Delta} \in PEF(\pi_m)} \sum_{\mathbf{d} \in \mathbf{\Delta}} \prod_{\mathbf{r} \in \mathbf{d}} P(\mathbf{r}) \qquad (2)$$

[3]Which differs from (Quirk and Menezes, 2006).

As usual, the parameters of this model are the PCFG rule probabilities which are estimated from the latent treebank using EM as explained next.

3.2 Learning Splits on Latent Treebank

For training the latent PCFG over the latent treebank, we resort to EM (Dempster et al., 1977) which estimates PCFG rule probabilities to maximize the likelihood of the parallel corpus instances. Computing expectations for EM is done efficiently using Inside-Outside (Lari and Young, 1990). As in other state splitting models (Matsuzaki et al., 2005), after splitting the non-terminals, we distribute the probability uniformly over the new rules, and we add to each new rule some random noise to break the symmetry. We split the non-terminals only once as in (Matsuzaki et al., 2005) (unlike (Petrov et al., 2006)). For estimating the distribution for unknown words we replace all words that appear ≤ 3 times with the "UNKNOWN" token.

3.3 Inference

We use CKY+ (Chappelier and Rajman, 1998) to parse a source sentence $\mathbf{s}$ into a forest using the learned split PCFG. Unfortunately, computing the most-likely permutation (or alternatively $\acute{\mathbf{s}}$) as in

$$\underset{\pi \in \Pi}{\operatorname{argmax}} \sum_{\mathbf{\Delta} \in PEF(\pi)} \sum_{\mathbf{d} \in \mathbf{\Delta}} P(\mathbf{d}, \pi_m)$$

from a lattice of permutations Π using a PCFG is NP-complete (Sima'an, 2002). Existing techniques, like variational decoding or Minimum-Bayes Risk (MBR), used for minimizing loss over trees as in (Petrov and Klein, 2007), are not directly applicable here. Hence, we opt for minimizing the risk of making an error under a loss function over permutations using the MBR decision rule (Kumar and Byrne, 2004):

$$\hat{\pi} = \underset{\pi}{\operatorname{argmin}} \sum_{\pi_r} Loss(\pi, \pi_r) P(\pi_r) \qquad (3)$$

The loss function we minimize is Kendall τ (Birch and Osborne, 2011; Isozaki et al., 2010a) which is a ratio of wrongly ordered pairs of words (including gapped pairs) to the total number of pairs. We do Monte Carlo sampling of 10000 derivations from the chart of the $\mathbf{s}$ and then find the least risky permutation in terms of this loss. We sample from the true distribution by sampling edges recursively using their inside probabilities. An empirical distribution over permutations $P(\pi)$ is given by the relative frequency of π in the sample.

With large samples it is hard to efficiently compute expected Kendall τ loss for each sampled hypothesis. For sentence of length k and sample of size n the complexity of a naive algorithm is $O(n^2k^2)$. Computing Kendall τ alone takes $O(k^2)$. We use the fact that Kendall τ decomposes as a linear function over all skip-bigrams b that could be built for any permutation of length k:

$$Kendall(\pi, \pi_r) = \sum_b \frac{1 - \delta(\pi, b)}{\frac{k(k-1)}{2}} \delta(\pi_r, b) \quad (4)$$

Here δ returns 1 if permutation π contains the skip bigram b, otherwise it returns 0. With this decomposition we can use the method from (DeNero et al., 2009) to efficiently compute the MBR hypothesis. Combining Equations 3 and 4 we get:

$$\hat{\pi} = \operatorname*{argmin}_{\pi} \sum_{\pi_r} \sum_b \frac{1 - \delta(\pi, b)}{\frac{k(k-1)}{2}} \delta(\pi_r, b) P(\pi_r) \quad (5)$$

We can move the summation inside and reformulate the expected Kendall τ loss as expectation over the skip-bigrams of the permutation.

$$= \operatorname*{argmin}_{\pi} \sum_b (1 - \delta(\pi, b)) \left[\sum_{\pi_r} \delta(\pi_r, b) P(\pi_r) \right] \quad (6)$$

$$= \operatorname*{argmin}_{\pi} \sum_b (1 - \delta(\pi, b)) \mathbb{E}_{P(\pi_r)} \delta(\pi_r, b) \quad (7)$$

$$= \operatorname*{argmax}_{\pi} \sum_b \delta(\pi, b) \mathbb{E}_{P(\pi_r)} \delta(\pi_r, b) \quad (8)$$

This means we need to pass through the sampled list only twice: (1) to compute expectations over skip bigrams and (2) to compute expected loss of each sampled permutation. The time complexity is $O(nk^2)$ which is quite fast in practice.

4 Experiments

We conduct experiments with three baselines:

- **Baseline A**: No preordering.
- **Baseline B**: Rule based preordering (Isozaki et al., 2010b), which first obtains an HPSG parse tree using Enju parser [4] and after that swaps the children by moving the syntactic head to the final position to account for different head orientation in English and Japanese.
- **Baseline C**: LADER (Neubig et al., 2012): latent variable preordering that is based on ITG and large-margin training with latent variables. We used LADER in standard settings without any linguistic features (POS tags or syntactic trees).

And we test four variants of our model:

- **RG$_{\text{left}}$** - only canonical left branching PET
- **RG$_{\text{right}}$** - only canonical right branching PET
- **RG$_{\text{ITG-forest}}$** - all PETs that are binary (ITG)
- **RG$_{\text{PET-forest}}$** - all PETs.

We test these models on English-Japanese NTCIR-8 Patent Translation (PATMT) Task. For tuning we use all NTCIR-7 dev sets and for testing the test set from NTCIR-9 from both directions. All used data was tokenized (English with Moses tokenizer and Japanese with KyTea [5]) and filtered for sentences between 4 and 50 words. A subset of this data is used for training the Reordering Grammar, obtained by filtering out sentences that have prime permutations of arity > 5, and for the ITG version arity > 2. Baseline C was trained on 600 sentences because training is prohibitively slow. Table 1 shows the sizes of data used.

corpus	#sents	#words source	#words target
train RG$_{\text{PET}}$	786k	21M	–
train RG$_{\text{ITG}}$	783k	21M	–
train LADER	600	15k	–
train translation	950k	25M	30M
tune translation	2k	55K	66K
test translation	3k	78K	93K

Table 1: Data stats

The Reordering Grammar was trained for 10 iterations of EM on *train RG* data. We use 30 splits for binary non-terminals and 3 for non-binary. Training on this dataset takes 2 days and parsing tuning and testing set without any pruning takes 11 and 18 hours respectively.

4.1 Intrinsic evaluation

We test how well our model predicts gold reorderings before translation by training the alignment model using MGIZA++ [6] on the training corpus and using it to align the test corpus. Gold reorderings for the test corpus are obtained by sorting words by their average target position and (unaligned words follow their right neighboring

[4]http://www.nactem.ac.uk/enju/

[5]http://www.phontron.com/kytea/

[6]http://www.kyloo.net/software/doku.php/mgiza:overview

word). We use Kendall τ score for evaluation (note the difference with Section 3.3 where we defined it as a loss function).

Table 2 shows that our models outperform all baselines on this task. The only strange result here is that rule-based preordering obtains a lower score than no preordering, which might be an artifact of the Enju parser changing the tokenization of its input, so the Kendall τ of this system might not really reflect the real quality of the preordering. All other systems use the same tokenization.

	Kendall τ
$A^{No\ preordering}$	0.7655
$B^{Rule\ based}$	0.7567
C^{LADER}	0.8176
$RG_{\text{left-branching}}$	0.8201
$RG_{\text{right-branching}}$	0.8246
$RG_{\text{ITG-forest}}$	0.823
$RG_{\text{PET-forest}}$	0.8255

Table 2: Reordering prediction

4.2 Extrinsic evaluation in MT

The reordered output of all the mentioned baselines and versions of our model are translated with phrase-based MT system (Koehn et al., 2007) (distortion limit set to 6 with distance based reordering model) that is trained on **gold preordering** of the training data [7] $\acute{\mathbf{s}} - \mathbf{t}$. The only exception is Baseline A which is trained on original $\mathbf{s} - \mathbf{t}$.

We use a 5-gram language model trained with KenLM [8], tune 3 times with kb-mira (Cherry and Foster, 2012) to account for tuner instability and evaluated using Multeval [9] for statistical significance on 3 metrics: BLEU (Papineni et al., 2002), METEOR (Denkowski and Lavie, 2014) and TER (Snover et al., 2006). We additionally report RIBES score (Isozaki et al., 2010a) that concentrates on word order more than other metrics.

Single or all PETs? In Table 3 we see that using *all PETs* during training makes a big impact on performance. Only the *all PETs* variants

System	BLEU ↑	METEOR ↑	TER ↓	RIBES ↑
$A^{No\ preord.}$	27.8	48.9	59.2	68.29
$B^{Rule\ based}$	29.6	48.7	59.2	71.12
C^{LADER}	31.1	50.5	56.0	74.29
RG_{left}	31.2^{AB}	50.5^{AB}	56.3^{AB}_{C}	**74.45**
RG_{right}	31.4^{AB}	50.5^{AB}	56.3^{AB}_{C}	**75.29**
$RG_{\text{ITG-forest}}$	$\mathbf{31.6^{ABC}}$	$\mathbf{50.8^{ABC}}$	$\mathbf{55.7^{ABC}}$	**75.29**
$RG_{\text{PET-forest}}$	$\mathbf{32.0^{ABC}}$	$\mathbf{51.0^{ABC}}$	$\mathbf{55.7^{ABC}}$	**75.62**

Table 3: Comparison of different preordering models. Superscripts A, B and C signify if the system is significantly better ($p < 0.05$) than the respective baseline or significantly worse (in which case it is a subscript). Significance tests were not computed for RIBES. Score is bold if the system is significantly better than all the baselines.

($RG_{\text{ITG-forest}}$ and $RG_{\text{PET-forest}}$) significantly outperform all baselines. If we are to choose a *single PET* per training instance, then learning RG from only left-branching PETs (the one usually chosen in other work, e.g. (Saluja et al., 2014)) performs slightly worse than the right-branching PET. This is possibly because English is mostly right-branching. So even though both PETs describe the same reordering, RG_{right} captures reordering over English input better than RG_{left}.

All PETs or binary only? *$RG_{\text{PET-forest}}$ performs significantly better than $RG_{\text{ITG-forest}}$* ($p < 0.05$). Non-ITG reordering operators are predicted rarely (in only 99 sentences of the test set), but they make a difference, because these operators often appear high in the predicted PET. Furthermore, having these operators during training might allow for better fit to the data.

How much reordering is resolved by the Reordering Grammar? Obviously, completely factorizing out the reordering from the translation process is impossible because reordering depends to a certain degree on target lexical choice. To quantify the contribution of Reordering Grammar, we tested decoding with different distortion limit values in the SMT system. We compare the phrase-based (PB) system with distance based cost function for reordering (Koehn et al., 2007) with and without preordering.

Figure 3 shows that Reordering Grammar gives substantial performance improvements at all distortion limits (both BLEU and RIBES). $RG_{\text{PET-forest}}$ is less sensitive to changes in decoder distortion limit than standard PBSMT. The perfor-

[7]Earlier work on preordering applies the preordering model to the training data to obtain a parallel corpus of *guessed* $\acute{\mathbf{s}} - \mathbf{t}$ pairs, which are the word re-aligned and then used for training the back-end MT system (Khalilov and Sima'an, 2011). We skip this, we take the risk of mismatch between the preordering and the back-end system, but this simplifies training and saves a good amount of training time.

[8]http://kheafield.com/code/kenlm/

[9]https://github.com/jhclark/multeval

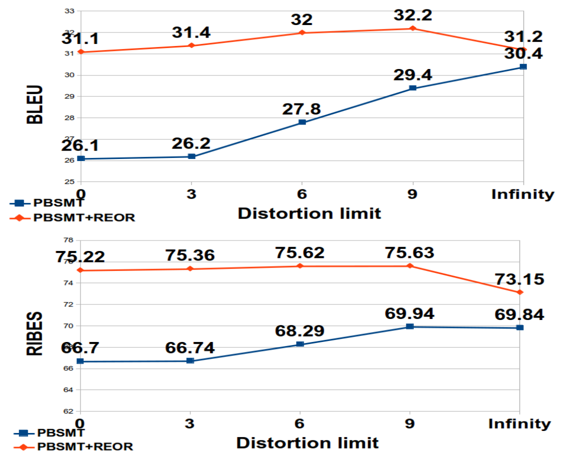


Figure 3: Distortion effect on BLEU and RIBES

mance of RG$_{\text{PET-forest}}$ varies only by 1.1 BLEU points while standard PBSMT by 4.3 BLEU points. Some local reordering in the decoder seems to help RG$_{\text{PET-forest}}$ but **large distortion limits seem to degrade the preordering choice**. This shows also that the improved performance of RG$_{\text{PET-forest}}$ is not only a result of efficiently exploring the full space of permutations, but also a result of improved scoring of permutations.

System	**BLEU** ↑	**METEOR** ↑	**TER** ↓	**RIBES** ↑
D^{PBMSD}	29.6	50.1	58.0	68.97
E^{Hiero}	32.6	52.1	54.5	74.12
RG$_{\text{PET-forest}}$+MSD	32.4^{D}	51.3$^{D}_{E}$	55.3$^{D}_{E}$	**75.72**

Table 4: Comparison to MSD and Hiero

Does the improvement remain for a decoder with MSD reordering model? We compare the RG$_{\text{PET-forest}}$ preordered model against a decoder that uses the strong MSD model (Tillmann, 2004; Koehn et al., 2007). Table 4 shows that using Reordering Grammar as front-end to MSD reordering (full Moses) improves performance by 2.8 BLEU points. The improvement is confirmed by METEOR, TER and RIBES. Our preordering model and MSD are complementary – the Reordering Grammar captures long distance reordering, while MSD possibly does better local reorderings, especially reorderings conditioned on the lexical part of translation units.

Interestingly, the MSD model (BLEU 29.6) improves over distance-based reordering (BLEU 27.8) by (BLEU 1.8), whereas the difference between these systems as back-ends to Reordering Grammar (respectively BLEU 32.4 and 32.0) is far smaller (0.4 BLEU). This suggests that a major share of reorderings can be handled well by preordering without conditioning on target lexical choice. Furthermore, this shows that RG$_{\text{PET-forest}}$ preordering is not very sensitive to the decoder's reordering model.

Comparison to a Hierarchical model (Hiero). Hierarchical preordering is not intended for a hierarchical model as Hiero (Chiang, 2005). Yet, here we compare our preordering system (PB MSD+RG) to Hiero for completeness, while we should keep in mind that Hiero's reordering model has access to much richer training data. We will discuss these differences shortly.

Table 4 shows that the difference in BLEU is not statistically significant, but there is more difference in METEOR and TER. RIBES, which concentrates more on reordering, prefers Reordering Grammar over Hiero. It is somewhat surprising that a preordering model combined with a phrase-based model succeeds to rival Hiero's performance on English-Japanese. Especially when looking at the differences between the two:

1. Reordering Grammar uses only minimal phrases, while Hiero uses composite (longer) phrases which encapsulate internal reorderings, but also non-contiguous phrases.
2. Hiero conditions its reordering on the lexical target side, whereas the Reordering Grammar does not (by definition).
3. Hiero uses a range of features, e.g., a language model, while Reordering Grammar is a mere generative PCFG.

The advantages of Hiero can be brought to bear upon Reordering Grammar by reformulating it as a discriminative model.

Which structure is learned? Figure 4 shows an example PET output showing how our model learns: (1) that the article "the" has no equivalent in Japanese, (2) that verbs go after their object, (3) to use postpositions instead of prepositions, and (4) to correctly group certain syntactic units, e.g. NPs and VPs.

5 Related work

The majority of work on preordering is based on syntactic parse trees, e.g., (Lerner and Petrov, 2013; Khalilov and Sima'an, 2011; Xia and Mccord, 2004). Here we concentrate on work that has common aspects with this work. Neubig et

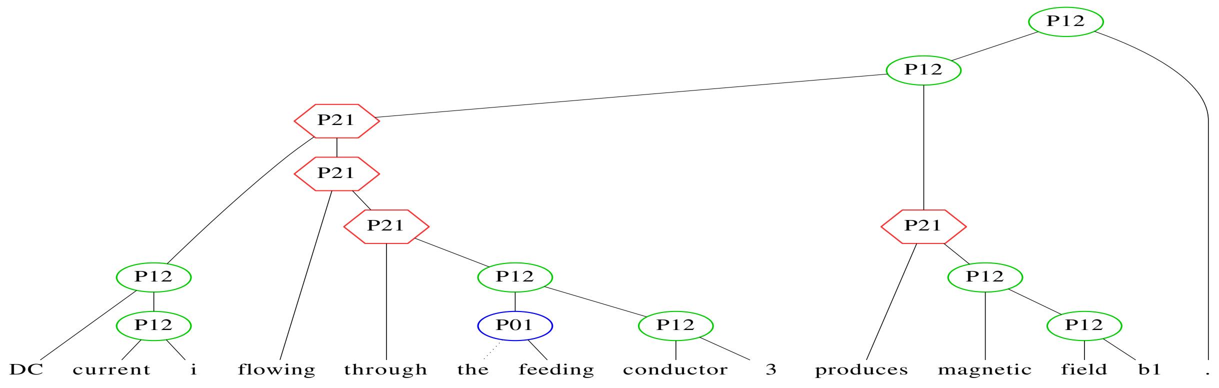


Figure 4: Example parse of English sentence that predicts reordering for English-Japanese

al (2012) trains a latent non-probabilistic discriminative model for preordering as an ITG-like grammar limited to binarizable permutations. Tromble and Eisner (2009) use ITG but do not train the grammar. They only use it to constrain the local search. DeNero and Uszkoreit (2011) present two separate consecutive steps for unsupervised induction of hierarchical structure (ITG) and the induction of a reordering function over it. In contrast, here we learn both the structure and the reordering function simultaneously. Furthermore, at test time, our inference with MBR over a measure of permutation (Kendall) allows exploiting both structure and reordering weights for inference, whereas test-time inference in (DeNero and Uszkoreit, 2011) is also a two step process – the parser forwards to the next stage the best parse.

Dyer and Resnik (2010) treat reordering as a latent variable and try to sum over all derivations that lead not only to the same reordering but also to the same translation. In their work they consider all permutations allowed by a given syntactic tree.

Saers et al (2012) induce synchronous grammar for translation by splitting the non-terminals, but unlike our approach they split generic nonterminals and not operators. Their most expressive grammar covers only binarizable permutations. The decoder that uses this model does not try to sum over many derivations that have the same yield. They do not make independence assumption like our "unary trick" which is probably the reason they do not split more than 8 times. They do not compare their results to any other SMT system and test on a very small dataset.

Saluja et al (2014) attempts inducing a refined Hiero grammar (latent synchronous CFG) from Normalized Decomposition Trees (NDT) (Zhang et al., 2008). While there are similarities with the present work, there are major differences. On the similarity side, NDTs are decomposing alignments in ways similar to PETs, and both Saluja's and our models refine the labels on the nodes of these decompositions. However, there are major differences between the two:

- Our model is completely monolingual and unlexicalized (does not condition its reordering on the translation) in contrast with the Latent SCFG used in (Saluja et al., 2014),
- Our Latent PCFG label splits are defined as refinements of prime permutations, i.e., specifically designed for learning reordering, whereas (Saluja et al., 2014) aims at learning label splitting that helps predicting NDTs from source sentences,
- Our model exploits **all PETs and all derivations**, both during training (latent treebank) and during inferences. In (Saluja et al., 2014) only left branching NDT derivations are used for learning the model.
- The training data used by (Saluja et al., 2014) is about 60 times smaller in number of words than the data used here; the test set of (Saluja et al., 2014) also consists of far shorter sentences where reordering could be less crucial.

A related work with a similar intuition is presented in (Maillette de Buy Wenniger and Sima'an, 2014), where nodes of a tree structure similar to PETs are labeled with reordering patterns obtained by factorizing word alignments into Hierarchical Alignment Trees. These patterns are used for labeling the standard Hiero grammar. Unlike this work, the labels extracted by (Maillette de Buy Wenniger and Sima'an, 2014) are clustered manually into less than a dozen labels without the possibility of fitting the labels to the training data.

6 Conclusion

We present a generative Reordering PCFG model learned from latent treebanks over PETs obtained by factorizing permutations over minimal phrase pairs. Our Reordering PCFG handles non-ITG reordering patterns (up to 5-ary branching) and it works with all PETs that factorize a permutation (rather than a single PET). To the best of our knowledge this is the first time both extensions are shown to improve performance. The empirical results on English-Japanese show that (1) when used for preordering, the Reordering PCFG helps particularly with relieving the phrase-based model from long range reorderings, (2) combined with a state-of-the-art phrase model, Reordering PCFG shows performance not too different from Hiero, supporting the common wisdom of factorizing long range reordering outside the decoder, (3) Reordering PCFG generates derivations that seem to coincide well with linguistically-motivated reordering patterns for English-Japanese. There are various direction we would like to explore, the most obvious of which are integrating the learned reordering with other feature functions in a discriminative setting, and extending the model to deal with non-contiguous minimal phrases.

Acknowledgments

This work is supported by STW grant nr. 12271 and NWO VICI grant nr. 277-89-002. We thank Wilker Aziz for comments on earlier version of the paper and discussions about MBR and sampling.

References

Michael H. Albert and Mike D. Atkinson. 2005. Simple Permutations and Pattern Restricted Permutations. *Discrete Mathematics*, 300(1-3):1–15.

Alexandra Birch and Miles Osborne. 2011. Reordering Metrics for MT. In *Proceedings of the Association for Computational Linguistics*, Portland, Oregon, USA. Association for Computational Linguistics.

Glenn Carroll and Mats Rooth. 1998. Valence Induction with a Head-Lexicalized PCFG. In *In Proceedings of Third Conference on Empirical Methods in Natural Language Processing*.

Jean-Cédric Chappelier and Martin Rajman. 1998. A Generalized CYK Algorithm for Parsing Stochastic CFG. In *Proceedings of the First Workshop on Tabulation in Parsing and Deduction*, pages 133–137.

Colin Cherry and George Foster. 2012. Batch Tuning Strategies for Statistical Machine Translation. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, NAACL HLT '12, pages 427–436.

David Chiang. 2005. A Hierarchical Phrase-Based Model for Statistical Machine Translation. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 263–270, Ann Arbor, Michigan, June.

Noam Chomsky. 1970. Remarks on Nominalization. In Roderick A. Jacobs and Peter S. Rosenbaum, editors, *Readings in English Transformational Grammar*, pages 184–221. Ginn, Boston.

Michael Collins, Philipp Koehn, and Ivona Kučerová. 2005. Clause Restructuring for Statistical Machine Translation. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, ACL '05, pages 531–540.

Michael Collins. 2003. Head-Driven Statistical Models for Natural Language Parsing. *Comput. Linguist.*, 29(4):589–637, December.

A. P. Dempster, N. M. Laird, and D. B. Rubin. 1977. Maximum Likelihood from Incomplete Data via the EM Algorithm. *JOURNAL OF THE ROYAL STATISTICAL SOCIETY, SERIES B*, 39(1):1–38.

John DeNero and Jakob Uszkoreit. 2011. Inducing Sentence Structure from Parallel Corpora for Reordering. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '11, pages 193–203.

John DeNero, David Chiang, and Kevin Knight. 2009. Fast Consensus Decoding over Translation Forests. 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: Volume 2 - Volume 2*, ACL '09, pages 567–575.

Michael Denkowski and Alon Lavie. 2014. Meteor Universal: Language Specific Translation Evaluation for Any Target Language. In *Proceedings of the EACL 2014 Workshop on Statistical Machine Translation*.

Chris Dyer and Philip Resnik. 2010. Context-free Reordering, Finite-state Translation. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, HLT '10, pages 858–866.

Hideki Isozaki, Tsutomu Hirao, Kevin Duh, Katsuhito Sudoh, and Hajime Tsukada. 2010a. Automatic Evaluation of Translation Quality for Distant Language Pairs. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, EMNLP '10, pages 944–952.

Hideki Isozaki, Katsuhito Sudoh, Hajime Tsukada, and Kevin Duh. 2010b. Head Finalization: A Simple Reordering Rule for SOV Languages. In *Proceedings of the Joint Fifth Workshop on Statistical Machine Translation and MetricsMATR*, WMT '10, pages 244–251.

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 *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 183–192, Edinburgh, Scotland, UK., July. Association for Computational Linguistics.

Maxim Khalilov and Khalil Sima'an. 2011. Context-Sensitive Syntactic Source-Reordering by Statistical Transduction. In *IJCNLP*, pages 38–46.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondřej Bojar, Alexandra Constantin, and Evan Herbst. 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*, ACL '07, pages 177–180.

Shankar Kumar and William Byrne. 2004. Minimum Bayes-Risk Decoding for Statistical Machine Translation. In *Proceedings of the Joint Conference on Human Language Technologies and the Annual Meeting of the North American Chapter of the Association of Computational Linguistics (HLT-NAACL)*.

K. Lari and S. J. Young. 1990. The Estimation of Stochastic Context-Free Grammars using the Inside-Outside Algorithm. *Computer Speech and Language*, 4:35–56.

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. Association for Computational Linguistics.

Gideon Maillette de Buy Wenniger and Khalil Sima'an. 2014. Bilingual Markov Reordering Labels for Hierarchical SMT. In *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*, pages 138–147, Doha, Qatar, October.

Takuya Matsuzaki, Yusuke Miyao, and Jun'ichi Tsujii. 2005. Probabilistic CFG with Latent Annotations. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, ACL '05, pages 75–82.

Graham Neubig, Taro Watanabe, and Shinsuke Mori. 2012. Inducing a Discriminative Parser to Optimize Machine Translation Reordering. In *Conference on Empirical Methods in Natural Language Processing and Natural Language Learning (EMNLP-CoNLL)*, pages 843–853, Jeju, Korea, July.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: A Method for Automatic Evaluation of Machine Translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ACL '02, pages 311–318.

Slav Petrov and Dan Klein. 2007. Improved Inference for Unlexicalized Parsing. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference*, pages 404–411, Rochester, New York, April.

Slav Petrov, Leon Barrett, Romain Thibaux, and Dan Klein. 2006. Learning Accurate, Compact, and Interpretable Tree Annotation. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 433–440, Sydney, Australia, July. Association for Computational Linguistics.

Detlef Prescher. 2005. Inducing Head-Driven PCFGs with Latent Heads: Refining a Tree-Bank Grammar for Parsing. In *In ECML05*.

Chris Quirk and Arul Menezes. 2006. Do we need phrases? Challenging the conventional wisdom in Statistical Machine Translation. In *Proceedings of HLT-NAACL 2006*. ACL/SIGPARSE, May.

Markus Saers, Karteek Addanki, and Dekai Wu. 2012. From Finite-State to Inversion Transductions: Toward Unsupervised Bilingual Grammar Induction. In *COLING 2012, 24th International Conference on Computational Linguistics, Proceedings of the Conference: Technical Papers, 8-15 December 2012, Mumbai, India*, pages 2325–2340.

A. Saluja, C. Dyer, and S. B. Cohen. 2014. Latent-Variable Synchronous CFGs for Hierarchical Translation. *Proceedings of EMNLP*.

Khalil Sima'an. 2002. Computational Complexity of Probabilistic Disambiguation. *Grammars*, 5(2):125–151.

Matthew Snover, Bonnie Dorr, Richard Schwartz, Linnea Micciulla, and John Makhoul. 2006. A study of translation edit rate with targeted human annotation. In *In Proceedings of Association for Machine Translation in the Americas*, pages 223–231.

Christoph Tillmann. 2004. A Unigram Orientation Model for Statistical Machine Translation. In *Proceedings of HLT-NAACL 2004: Short Papers*, HLT-NAACL-Short '04, pages 101–104.

Roy Tromble and Jason Eisner. 2009. Learning Linear Ordering Problems for Better Translation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1007–1016, Singapore, August.

Benjamin Wellington, Sonjia Waxmonsky, and I. Dan Melamed. 2006. Empirical lower bounds on the complexity of translational equivalence. In *In Proceedings of ACL 2006*, pages 977–984.

Dekai Wu. 1997. Stochastic Inversion Transduction Grammars and Bilingual Parsing of Parallel Corpora. *Comput. Linguist.*, 23(3):377–403, September.

Fei Xia and Michael Mccord. 2004. Improving a Statistical MT System with Automatically Learned Rewrite Patterns. In *Proceedings of Coling 2004*, pages 508–514, Geneva, Switzerland, August. COLING.

Hao Zhang and Daniel Gildea. 2007. Factorization of Synchronous Context-Free Grammars in Linear Time. In *NAACL Workshop on Syntax and Structure in Statistical Translation (SSST)*, pages 25–32.

Hao Zhang, Daniel Gildea, and David Chiang. 2008. Extracting Synchronous Grammar Rules From Word-Level Alignments in Linear Time. In *Proceedings of the 22nd International Conference on Computational Linguistics (COLING-08)*, pages 1081–1088, Manchester, UK.