Finding Reordering Grammar in Hidden Treebanks

Miloš Stanojević and Khalil Sima'an
Institute for Logic, Language and Computation
University of Amsterdam
m.stanojevic@uva.nl and k.simaan@uva.nl

May 12, 2015

Abstract

We present a novel algorithm 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 tested on an intrinsic task (predicting word order before translation) and extrinsic task (improving the quality of the final translation). The performance of our preordering approach for phase-based models gives large gains over vanilla phrase reordering and is comparable to that of Hiero on 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 some 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].

The two steps of preordering are: finding a tree suitable for transduction, and then finding the 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]. While we believe in the advantages that trees offer for long range reordering cf. [Chiang, 2005], we realize that limiting reordering to sibling syntactic constituents can be sub-optimal because the reordered source spans are not always syntactic constituents. 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.

Learning both trees and transductions jointly raises two questions. How to obtain suitable trees for the source sentence? And how to learn 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].

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.

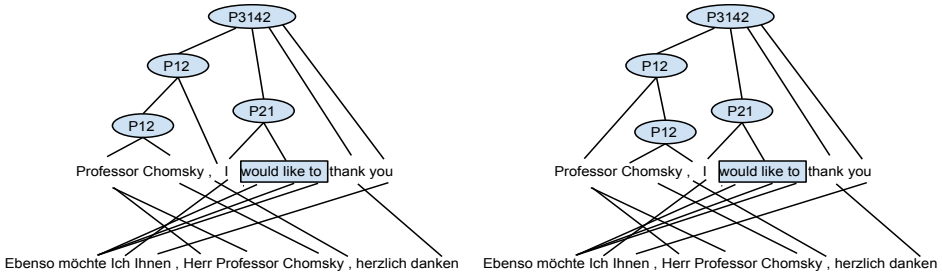
Figure 1 shows two alternative PETs for the same permutation over minimal phrases. The nodes have labels (like $P3142$) which stand for local permutations (called prime permutation) over the child nodes; for example, the root label $P3142$ stands for prime permutation $\langle 3, 1, 4, 2\rangle$, which 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 define exactly the set of 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 training permutations in terms of both phrases and their reordering.

For learning preordering, we start out by extracting an *initial* PCFG from our 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 in the treebank. 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 ($\mathbf{s}$) instead of a single tree.

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 $\mathbf{s}$, we map the splits back on their initial prime permutations, and then retrieve a reordered version $\acute{\mathbf{s}}$ of $\mathbf{s}$. In this sense, our latent splits are *dedicated to reordering*.

There are two technical difficulties alien to work on latent PCFGs in treebank parsing. Firstly, as mentioned above, permutations may factorize into more



(a) Canonical PET (b) Alternative PET

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

than one PET (a forest) leading to a latent training treebank.[1] And secondly, after we parse a source string $\mathbf{s}$, we are interested in $\acute{\mathbf{s}}$, the permuted version of $\mathbf{s}$, not in the best derivation/PET. Exact computation is a known NP-Complete problem [Sima'an, 2002]. We tackle this with sampling derivations and Minimum-Bayes Risk decoding, where we use Kendall reordering score as loss function, which is an efficient measure over permutations [Birch and Osborne, 2011, Isozaki et al., 2010].

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 $\acute{\mathbf{s}}$.

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. In fact, our model bridges the performance gap with Hiero [Chiang, 2005], while our model does not use any lexicalized reordering rules and does not condition on the target lexical choice.

2 PETs and the Hidden Treebank

We aim at learning a PCFG which we will use for parsing source sentences $\mathbf{s}$ into synchronous trees, from which we can obtain a reordered source version $\acute{\mathbf{s}}$. 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 $\mathbf{s}$ and $\acute{\mathbf{s}}$ are over the same alphabet.

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

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 $P12$, $P3142$ and $P21$ 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 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.

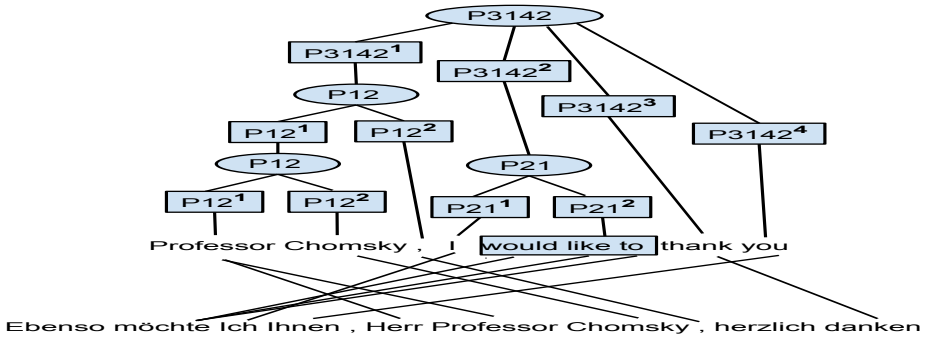


Figure 2: Permutation Tree with unary trick

3 Details of Latent Reordering PCFG

Obtaining permutations Let be given a source sentence $\mathbf{s}$ and its alignment $\mathbf{a}$ to a target sentence $\mathbf{t}$ in the training corpus. We segment the triple $\langle \mathbf{s}, \mathbf{a}, \mathbf{t} \rangle$ into a sequence of *minimal phrases* $\mathbf{s}_m$ (maximal sequence) such that the reordering between these minimal phrases constitutes a permutation π_m (they might not be minimal in the sense used in [Quirk and Menezes, 2006]). 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.

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

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 $\mathbf{d}$:

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

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

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

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 our latent PCFG over the latent treebank, we resort to the EM algorithm [Dempster et al., 1977]. EM estimates the probability of the PCFG rules that maximize the likelihood of the parallel corpus instances. Computing expectations for EM can be done efficiently using Inside-Outside [Lari and Young, 1990]. As in other state splitting models [Matsuzaki et al., 2005], after splitting the nonterminals, 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 nonterminals only once as in [Matsuzaki et al., 2005] (unlike [Petrov et al., 2006]). For estimating unknown words we apply the standard technique of replacing all

words that appear ≤ 3 times with the "UNKNOWN" token and then during parsing we use the distribution of rules over this token for all the words in the test sentences that we have not seen previously in the training data.

3.3 Inference

We use CKY+ [Chappelier and Rajman, 1998] to parse a $\mathbf{s}$ into a forest using the learned split PCFG. Unfortunately, computing the most-likely permutation (or alternatively $\hat{\mathbf{s}}$), $\text{argmax}_\pi \sum_{\mathbf{\Delta} \in PEF(\pi)} \sum_{\mathbf{d} \in \mathbf{\Delta}} P(\mathbf{d}, \pi_m)$, from a (sausage) lattice of possible permutations with a PCFG, is NP-complete [Sima'an, 2002]. Existing parsing techniques, like variational decoding or Minimum-Bayes Risk (MBR), used for minimizing loss over trees as those used 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]:

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

The loss function we minimize is Kendall τ [Birch and Osborne, 2011, Isozaki et al., 2010] 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 [Finkel et al., 2006]. The probability $P(\pi)$ is determined by relative frequency of π in the sample.

With big 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:

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

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

$$\hat{\pi} = \underset{\pi}{\text{argmin}} \sum_{\pi_r} \sum_b \frac{1 - \delta(\pi, b)}{\frac{k(k-1)}{2}} \delta(\pi_r, b) P(\pi_r) \tag{5}$$

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

$$= \underset{\pi}{\text{argmin}} \sum_b (1 - \delta(\pi, b)) \Big[\textstyle\sum_{\pi_r} \delta(\pi_r, b) P(\pi_r) \Big] \tag{6}$$

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

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

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 our experiments on English-Japanese with the dataset from NTCIR-8 Patent Translation (PATMT) Task. As tuning corpus we used all NTCIR-7 dev sets. For testing we used test set from NTCIR-9 from both directions. All used data was tokenized (English with Moses tokenizer and Japanese with KyTea [3]) and filtered for sentences between length of 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 with arity larger than 5. Table 1 shows the sizes of data used.

corpus	#sents	#words source	#words target
train reordering	786k	21M	–
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 reordering* 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. Figure 3 shows the perplexities after each iteration.

	Kendall τ score
No reordering	0.766
Reordering Grammar	0.826

Table 2: Reordering prediction

To test how well our model predicts gold reorderings before translation we trained the alignment model using MGIZA++ [4] on the training corpus and used it

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

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

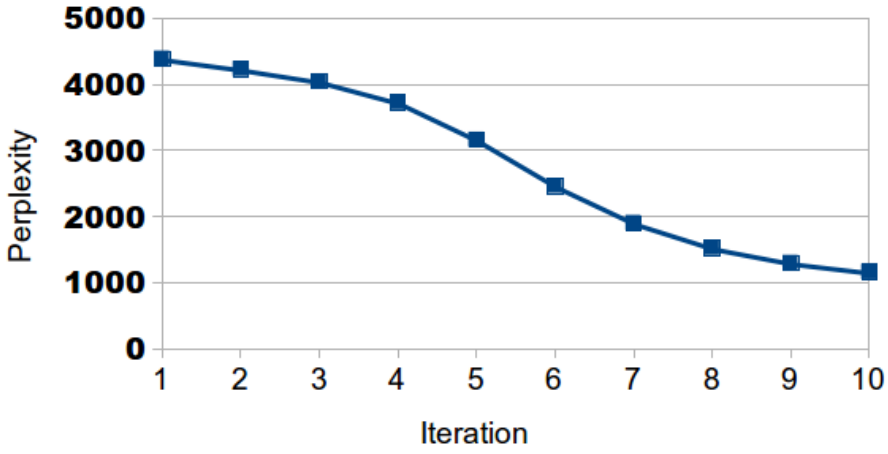


Figure 3: Perplexity on the training data

to align the testing corpus. Gold reorderings for test corpus is obtained by sorting words by their average target position and (unaligned words follow their right neighboring word). The results on predicting this word order are shown in Table 2. We use Kendall τ score for evaluation (note the difference with Section 3.3 where we defined it as a loss function).

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. The system that uses preordering is trained on **gold preordering** of the training data[5] $\acute{\mathbf{s}} - \mathbf{t}$. We use a 5-gram language model trained with KenLM [6], tune 3 times with kb-mira [Cherry and Foster, 2012] to account for tuner instability and evaluated using Multeval [7] for statistical significance on 3 metrics: BLEU [Papineni et al., 2002], METEOR [Denkowski and Lavie, 2014] and TER [Snover et al., 2006].

As it can be seen from Figures 4, 5 and 6, judged by all metrics, the Reordering

[5]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.

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

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

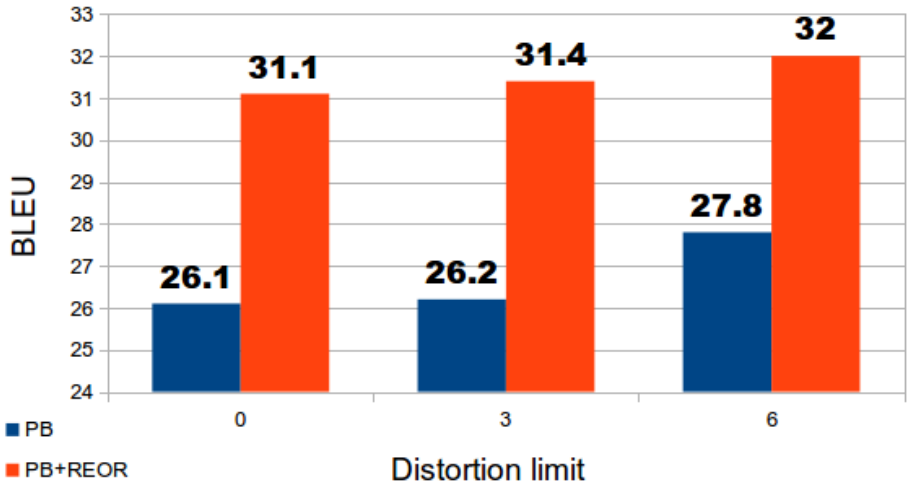


Figure 4: Distortion effect on BLEU

Grammar gives substantial performance improvements at all distortion limits. In fact, by increasing the distortion limit the performance increase is rather small relative to the improvement that preordering gives. Our preordering model improves the baseline for 5 BLEU points on monotone decoding, 4.2 BLEU on standard phrase based baseline with distortion limit set to 6. This suggests that the Reordering Grammar is relieving the back-end MT system from difficult reorderings it cannot usually resolve even with longer distortion limit.

Comparison to MSD reordering Now we experiment with a stronger baseline which works with phrase orientations – MSD (Monotone, Swap, Discontinuous) [Tillmann, 2004, Koehn et al., 2007]. Will the preordering model maintain its performance improvement? The results are shown in Table 3. Using Reordering Grammar as front-end to MSD reordering (full Moses baseline) improves performance by 2.8 BLEU points. The improvement is confirmed by METEOR and TER; in all cases it is statistically significant. 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 the translation unit.

One interesting observation is that MSD model (BLEU 29.6) improvement over distance-based reordering (BLEU 27.8) is relatively large (BLEU 1.8), whereas the difference between these systems as back-end to the Reordering Grammar (respectively BLEU 32.4 and 32.0) is far smaller (0.4 BLEU). Again, this suggests that our preordering model already handles a major share of the reorderings usually handles inside the decoder, i.e., it shows that this share of reorderings could be often handled well without conditioning on target lexical choice.

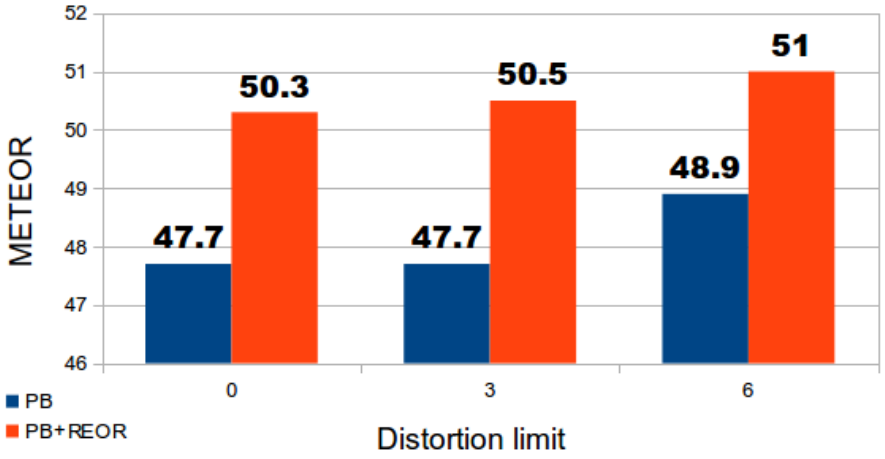


Figure 5: Distortion effect on METEOR

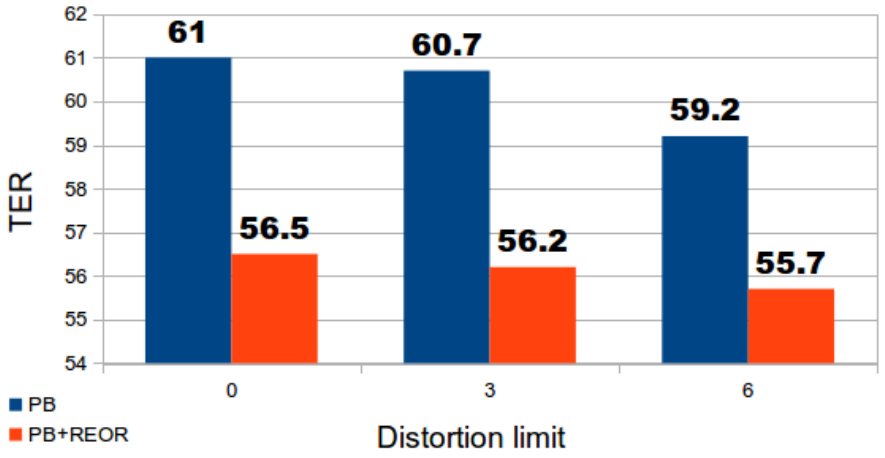


Figure 6: Distortion effect on TER

Metric	System	Avg	p-value
BLEU ↑	PB MSD	29.6	-
	PB MSD + REOR	32.4	0.00
METEOR ↑	PB MSD	50.1	-
	PB MSD + REOR	51.3	0.00
TER ↓	PB MSD	58.0	-
	PB MSD + REOR	55.3	0.00

Table 3: Phrase-Based MSD with/out preordering

Metric	System	Avg	p-value
BLEU ↑	Hiero	32.6	-
	PB MSD + REOR	32.4	0.16
METEOR ↑	Hiero	52.1	-
	PB MSD + REOR	51.3	0.00
TER ↓	Hiero	54.5	-
	PB MSD + REOR	55.3	0.00

Table 4: PB MSD+Preordering vs Hiero

Comparison to Hiero Although hierarchical preordering is not intended to be combined with Hiero [Chiang, 2005], as it is already a hierarchical model, it is interesting to compare performance. Note that Hiero's reordering model has access to much richer training data since it includes the target side. We will discuss these differences shortly after we discuss the results.

We compare our preordering system (PB MSD+REOR) to Hiero in Table 4. We see that the difference in BLEU is not statistically significant, but there is more difference in METEOR and TER. It is somewhat surprising that a preordering model combined with a standard phrase-based model succeeds to rival Hiero's performance on English-Japanese. Especially when looking at the differences between the two:

1. Our Reordering Grammar uses only minimal phrases, while Hiero uses phrases with gaps. Thus, 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). Hence, the preordering decisions are completely monolingual and unlexicalized,
3. Hiero uses a range of features, for example a language model, while the preordering grammar is a mere PCFG (generative without additional features),

Note that the advantages of Hiero can be brought to bear upon our preordering model by reformulating our model into a discriminative model with extra features. Since our model is essentially a monolingual parsing model, all known monolingual parsing techniques could be applied, e.g., discriminative reranking [Charniak and Johnson, 2005].

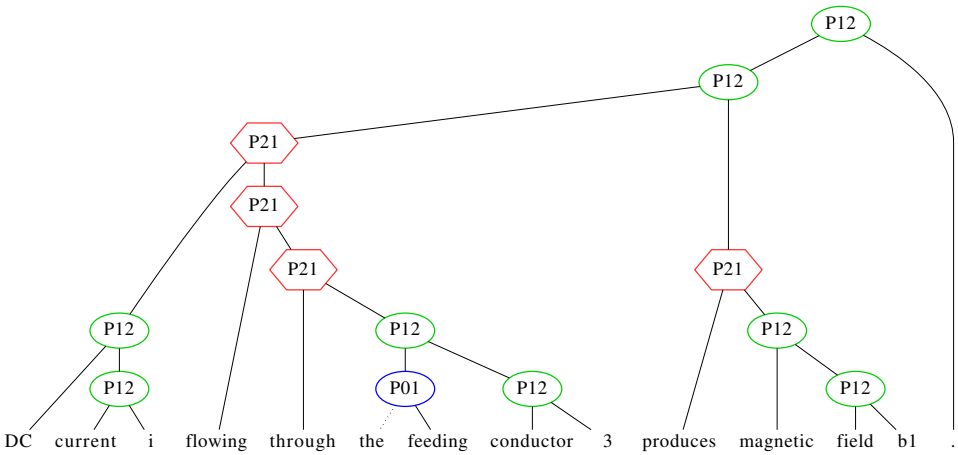


Figure 7: Example parse for English-Japanese

Example output Figure 7 shows an example preordered output which exemplifies how our model learns correctly:

- that the article "the" does not have an equivalent in Japanese,
- that verbs go after their object which is equivalent to switching head-directionality parameter [Chomsky, 1970]
- to use postpositions instead of prepositions (again, this is an effect of the head parameter)
- to correctly group certain syntactic units (in this example 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 al neubig trains a latent non-probabilistic discriminative model for preordering as an ITG-like grammar limited to binarizable permutations. Tromble and Eisner tromble use ITG but do not train the grammar. They only use it to constrain the local search. DeNero and Uszkoreit uszkoreit 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 dyerReodering 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. Other contributions of [Dyer and Resnik, 2010], such as the composition with finite-state translation model, are complementary with the present work and could give further improvements. Saluja et al cohen 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.
- On the practical side, the training data used by [Saluja et al., 2014] is about 60 times smaller in size (number of words) than the data used in the present work; the test set of [Saluja et al., 2014] also consists of far shorter sentences where reordering could be less of an issue.

6 Conclusion

We present a generative Reordering PCFG model learned from latent treebanks over PETs obtained by factorizing permutations over minimal phrase pairs. We use prime permutations on the PET nodes as initial reordering nonterminals of a monolingual source PCFG, which split to fit the permutations in a training parallel corpus. Our empirical results on reasonably sized English-Japanese data show that (1) When used as preordering, our model exhibits major performance improvement over a state-of-the-art phrase-based back-end system. Our analysis shows that the Reordering PCFG helps particularly with relieving long range reorderings which the back-end system cannot resolve otherwise, (2) When the preordering model is combined with the state-of-the-art phrase-based back-end its performance is not too different from that of Hiero system which exploits resources for hierarchical reordering not accessible for our model, (3) The reordering PCFG generates derivations that seem to coincide well with the reordering patterns that are linguistically expected for English-Japanese translation. There are many extension we would like to explore but the most obvious are integrating the learned reordering with other feature functions in a discriminative setting, and extending the model to

deal with non-contiguous minimal phrases.

References

[Albert and Atkinson, 2005] Albert, M. H. and Atkinson, M. D. (2005). Simple permutations and pattern restricted permutations. *Discrete Mathematics*, 300(1-3):1–15.

[Birch and Osborne, 2011] Birch, A. and Osborne, M. (2011). Reordering Metrics for MT. In *Proceedings of the Association for Computational Linguistics*, Portland, Oregon, USA. Association for Computational Linguistics.

[Carroll and Rooth, 1998] Carroll, G. and Rooth, M. (1998). Valence induction with a head-lexicalized PCFG. In *In Proceedings of Third Conference on Empirical Methods in Natural Language Processing*.

[Chappelier and Rajman, 1998] Chappelier, J.-C. and Rajman, M. (1998). A Generalized CYK Algorithm for Parsing Stochastic CFG. In *Proceedings of the First Workshop on Tabulation in Parsing and Deduction*, pages 133–137.

[Charniak and Johnson, 2005] Charniak, E. and Johnson, M. (2005). Coarse-to-fine N-best Parsing and MaxEnt Discriminative Reranking. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, ACL '05, pages 173–180, Stroudsburg, PA, USA. Association for Computational Linguistics.

[Cherry and Foster, 2012] Cherry, C. and Foster, G. (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, Stroudsburg, PA, USA. Association for Computational Linguistics.

[Chiang, 2005] Chiang, D. (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. Association for Computational Linguistics.

[Chomsky, 1970] Chomsky, N. (1970). Remarks on Nominalization. In Jacobs, R. A. and Rosenbaum, P. S., editors, *Readings in English Transformational Grammar*, pages 184–221. Ginn, Boston.

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

[Collins et al., 2005] Collins, M., Koehn, P., and Kučerová, I. (2005). Clause Restructuring for Statistical Machine Translation. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, ACL '05, pages 531–540, Stroudsburg, PA, USA. Association for Computational Linguistics.

[Dempster et al., 1977] Dempster, A. P., Laird, N. M., and Rubin, D. B. (1977). Maximum likelihood from incomplete data via the EM algorithm. *JOURNAL OF THE ROYAL STATISTICAL SOCIETY, SERIES B*, 39(1):1–38.

[DeNero et al., 2009] DeNero, J., Chiang, D., and Knight, K. (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, Stroudsburg, PA, USA. Association for Computational Linguistics.

[DeNero and Uszkoreit, 2011] DeNero, J. and Uszkoreit, J. (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, Stroudsburg, PA, USA. Association for Computational Linguistics.

[Denkowski and Lavie, 2014] Denkowski, M. and Lavie, A. (2014). Meteor Universal: Language Specific Translation Evaluation for Any Target Language. In *Proceedings of the EACL 2014 Workshop on Statistical Machine Translation*.

[Dyer and Resnik, 2010] Dyer, C. and Resnik, P. (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, Stroudsburg, PA, USA. Association for Computational Linguistics.

[Finkel et al., 2006] Finkel, J. R., Manning, C. D., and Ng, A. Y. (2006). Solving the Problem of Cascading Errors: Approximate Bayesian Inference for Linguistic Annotation Pipelines. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, EMNLP '06, pages 618–626, Stroudsburg, PA, USA. Association for Computational Linguistics.

[Isozaki et al., 2010] Isozaki, H., Hirao, T., Duh, K., Sudoh, K., and Tsukada, H. (2010). 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, Stroudsburg, PA, USA. Association for Computational Linguistics.

[Katz-Brown et al., 2011] Katz-Brown, J., Petrov, S., McDonald, R., Och, F., Talbot, D., Ichikawa, H., Seno, M., and Kazawa, H. (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. Association for Computational Linguistics.

[Khalilov and Sima'an, 2011] Khalilov, M. and Sima'an, K. (2011). Context-Sensitive Syntactic Source-Reordering by Statistical Transduction. In *IJCNLP*, pages 38–46.

[Koehn et al., 2007] Koehn, P., Hoang, H., Birch, A., Callison-Burch, C., Federico, M., Bertoldi, N., Cowan, B., Shen, W., Moran, C., Zens, R., Dyer, C., Bojar, O., Constantin, A., and Herbst, E. (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, Stroudsburg, PA, USA. Association for Computational Linguistics.

[Kumar and Byrne, 2004] Kumar, S. and Byrne, W. (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)*.

[Lari and Young, 1990] Lari, K. and Young, S. J. (1990). The estimation of stochastic context-free grammars using the inside-outside algorithm. *Computer Speech and Language*, 4:35–56.

[Lerner and Petrov, 2013] Lerner, U. and Petrov, S. (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.

[Matsuzaki et al., 2005] Matsuzaki, T., Miyao, Y., and Tsujii, J. (2005). Probabilistic CFG with Latent Annotations. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, ACL '05, pages 75–82, Stroudsburg, PA, USA. Association for Computational Linguistics.

[Papineni et al., 2002] Papineni, K., Roukos, S., Ward, T., and Zhu, W.-J. (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, Stroudsburg, PA, USA. Association for Computational Linguistics.

[Petrov et al., 2006] Petrov, S., Barrett, L., Thibaux, R., and Klein, D. (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. Association for Computational Linguistics.

[Petrov and Klein, 2007] Petrov, S. and Klein, D. (2007). Improved Inference for Unlexicalized Parsing. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Lin-*

guistics; Proceedings of the Main Conference, pages 404–411, Rochester, New York. Association for Computational Linguistics.

[Prescher, 2005] Prescher, D. (2005). Inducing head-driven PCFGs with latent heads: Refining a tree-bank grammar for parsing. In *In ECML05*.

[Quirk and Menezes, 2006] Quirk, C. and Menezes, A. (2006). Do we need phrases? Challenging the conventional wisdom in Statistical Machine Translation. In *Proceedings of HLT-NAACL 2006*. ACL/SIGPARSE.

[Saluja et al., 2014] Saluja, A., Dyer, C., and Cohen, S. B. (2014). Latent-variable synchronous CFGs for hierarchical translation. *Proceedings of EMNLP*.

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

[Snover et al., 2006] Snover, M., Dorr, B., Schwartz, R., Micciulla, L., and Makhoul, J. (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.

[Tillmann, 2004] Tillmann, C. (2004). A Unigram Orientation Model for Statistical Machine Translation. In *Proceedings of HLT-NAACL 2004: Short Papers*, HLT-NAACL-Short '04, pages 101–104, Stroudsburg, PA, USA. Association for Computational Linguistics.

[Wellington et al., 2006] Wellington, B., Waxmonsky, S., and Melamed, I. D. (2006). Empirical lower bounds on the complexity of translational equivalence. In *In Proceedings of ACL 2006*, pages 977–984.

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

[Xia and Mccord, 2004] Xia, F. and Mccord, M. (2004). Improving a Statistical MT System with Automatically Learned Rewrite Patterns. In *Proceedings of Coling 2004*, pages 508–514, Geneva, Switzerland. COLING.

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

[Zhang et al., 2008] Zhang, H., Gildea, D., and Chiang, D. (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.