Hierarchical Alignment Trees: A Recursive Factorization of Reordering in Word Alignments with Empirical Results

Khalil Sima'an[*]
Gideon Maillette de Buy Wenniger[†]

Abstract

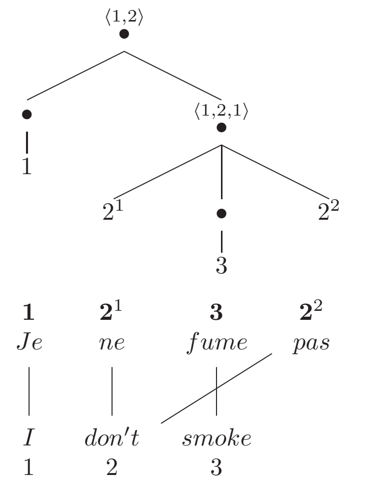
A word alignment in machine translation (MT) usually represents a mapping between word positions in a source sentence and word positions in its target language translation. In current MT research, word alignments are interpreted as grouping constrains usually exploited to define translation equivalence relations between source and target phrases. Existing work [Zhang et al., 2008] has endeavored to compactly and recursively represent the translation equivalence relations (phrase pairs) defined by an arbitrary word alignment. However, the problem of factorizing the *reordering aspects* of word alignments into recursive tree-like representations has received little attention in the literature. In the present paper we contribute a polynomial-time algorithm for factorizing any given word alignment into a decorated tree representation (called Hierarchical Alignment Tree – HAT) such that the word alignment is *uniquely* determined by its HAT representation. A HAT is the maximal tree for its word alignment and every one of its internal nodes is decorated with a local transduction operator. The transduction operators provide an explicit, recursive factorization of reordering/distortion patterns in word alignments, thereby generalizing a range of existing work on factorizing permutations (the right-hand sides of synchronous grammar rules). As a first application of the HAT algorithm, we contribute an empirical study revealing a *break-down* into subclasses of HATs that are equivalent to subclasses of manual as well as automatic word alignments. We find that, on average, manual word alignments have far shallower HATs than their automatically constructed counterparts, but that both exhibit surprisingly many more complex alignment phenomena than reported by earlier work. Our empirical study highlights one application of HATs as a tool for analyzing and visualizing the recursive reordering properties of word alignments.

1 Introduction

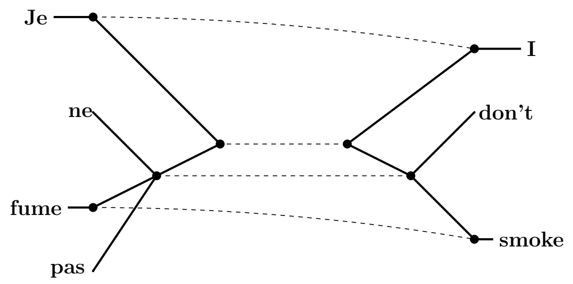
A major challenge for hierarchical machine translation (MT) is to systematically define for every source-target sentence pair in a parallel corpus bilingual recursive structures that show how the target-language translation of the source sentence is built up from the translations of its parts. The core of this challenge is to align together, recursively, the parts that are translation equivalents in every sentence pair [Wu, 1996, Wu, 1997, Wu and Wong, 1998]. Recursive alignment at the sub-sentential level (in contrast with the word level) is often represented as a synchronous tree pair (STP), i.e., source and target trees with alignment links between their nodes (see Figure 1). Nodes that are linked together dominate fringes that are considered translation equivalents. Like word alignments [Brown et al., 1990, Och and Ney, 2003], node alignments in STPs show the reordering patterns, albeit in a recursively structured fashion, which is the main focus of this article.

There are various possible approaches to put STPs into parallel corpora. One approach is to parse the source and/or target sentences, and then align nodes in the syntactic trees on both sides, e.g., [Poutsma, 2000, Tinsley et al., 2009, Zhechev and Way, 2008]. Because syntactic trees do not always coincide with accepted notions of translation equivalence (e.g., non-constituent ngram pairs like the well-known phrase

[*]Institute for Logic, Language and Computation, University of Amsterdam. k.simaan@uva.nl
[†]Institute for Logic, Language and Computation, University of Amsterdam. gemdb@gmail.com



(a) Bottom: A word aligned sentence pair. Top: The HAT representation.



(b) An STP representation showing the node-aligned pair of trees that the HAT in Figure 1a packs together.

Figure 1: In Figure 1a, the word-aligned sentence pair is shown with the French words decorated with the linked English positions (e.g., `Je` and `fume` are aligned respectively with English positions 1 and 3). We use extra superscript notation (as in 2^1 and 2^2) to represent the fact that French words `ne` and `pas` are first and second parts of a construction that aligns with English word in position 2 (`don't`). Figure 1a also shows the HAT representation. The notation $\langle 1, 2, 1 \rangle$ on the node dominating the `ne fume pas` construction stands for a *transduction operator* which links the first and third children of this node (respectively dominating `ne` and `pas`) with the first child under the corresponding node on the English side; similarly the 2 in $\langle 1, 2, 1 \rangle$ links the second child of this node with the second child of the corresponding node on the English side.

1

pairs), it is often necessary to resort to modifications in the structural aspects of the syntactic trees, e.g., binarizing n-ary rules, flattening deep structures and relabeling certain nodes, cf. [Wang et al., 2010]. An alternative statistical approach aims directly at inducing STPs as hidden components of a translation model, which coincides with the original ideas of, e.g., [Wu, 1996, Wu and Wong, 1998]. Inducing STPs into parallel corpora is a relatively new research topic, e.g., [Eisner, 2003, Blunsom et al., 2008], hindered by computational and statistical requirements. A third approach is to employ standard word alignments as the starting point for putting STPs over string pairs in parallel corpora. The word alignments can be used for constraining the induction process (which could constrain the word alignment in turn), e.g., [Haghighi et al., 2009, Mylonakis and Sima'an, 2010]. The latter entails the need for a rigorous approach to *interpret word alignments as constraints over STPs*. But, interpreting word alignments as constraints over STPs also suggests that word alignments themselves could be viewed as defining STPs of a certain family.

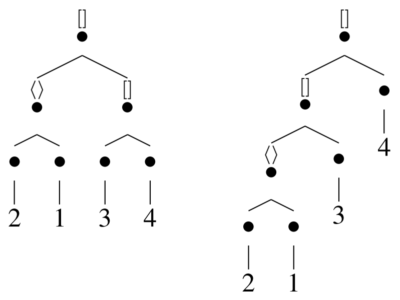


Figure 2: The two BITTs for permutation $\langle 2, 1, 3, 4 \rangle$. Bullets dominating terminal nodes represent paired units of translation equivalence.

In this article we are concerned with the problem of devising an algorithm for interpreting word alignments as tree representations (akin to STPs) and empirically studying the recursive reordering properties of some of the commonly used word alignments in parallel corpora. In fact, there exists related work [Zhang et al., 2008] on the first part of this general problem: they present a *compact tree representation (called Normalized Decomposition Trees – NDTs)* that allows efficient extraction of all phrase pairs defined by a word alignment. Zhang et al. also discuss a related sub-problem concerning factorizing Synchronous Context-Free Grammar (SCFG) rules (*strict permutations*) for more efficient parsing [Zhang et al., 2006, Huang et al., 2009, Gildea et al., 2006]. Our work is strongly related to this range of earlier work, and it can been seen as generalizing the existing work on factorizing permutations. While clearly related to [Zhang et al., 2008], the present work differs in its main goals and the resulting representation: we aim for a tree-like representation that is focused on the reordering aspect (as opposed to phrase pairs) of word alignments and that retains these aspects of word alignments (i.e., without loss of reordering information). More accurately, the current work starts out from different fundamental requirements for hierarchically representing word alignments with tree structures (called Hierarchical Alignment Trees – HATs) than previous work:

Uniqueness A HAT defines a word alignment uniquely.

Maximal A HAT is a maximal tree that factorizes the word alignment recursively. For every word alignment this can be achieved by selecting the minimum allowed branching factor at every node in the tree, cf. earlier work on factorizing SCFG rules [Zhang et al., 2006, Huang et al., 2009, Gildea et al., 2006, Zhang et al., 2008]. Like [Zhang et al., 2008] we also assume that fully lexicalized translation equivalence units are restricted to phrase pairs.

Node operators The HAT representation explicitly shows the recursive reordering in a word alignment using *transduction operators* decorating every sub-sentential level (node), in analogy to the way

Binary-branching Inversion-Transduction Trees [Wu, 1996] explicitly show the inversion ($<>$) and monotone ($[]$) operators on every node (see Figure 2).

Figure 1a shows an example word alignment and its HAT. The HAT shows the local transduction operators on the nodes (see explanation in caption of Figure 1). We think that such explicit node operators could be useful, e.g., for statistical learning of reordering models, or for studying the hierarchical properties of word alignments.[1]

On the empirical side, we aim for HATs that help us *bucket word alignments into sub-families* according to hierarchical reordering properties. The empirical part of this article (Section 6) exemplifies this by providing a detailed study of manual and automatic word alignments. HATs, with transduction operators on every node, are shown to help in distinguishing between various types of word alignments: those that make use only of the pair of ITG (Inversion Transduction Grammar) operators, others that use general permutations, and finally the most general case which goes beyond permutations over integers.

Finally, one of the original goals that lead to this work is to *visualize* the hierarchical structure of word alignments to help human qualitative analysis of how word alignments could diverge from (or coincide with) monolingual syntactic structure. We think that the HATs presented in this work are suitable for this goal because they can be visualized as STPs where source and target HATs are node-linked but retaining the node transduction operators. One plus point for HATs in this regard is that HATs *explicitly* show the *non-contiguous* parts of translation equivalence units, e.g., Figure 1a shows directly that the non-contiguous French negation `ne ... pas` is equivalent to English `not`. The nodes in the HATs correspond to the usual phrase pairs [Chiang, 2007], yet the HAT representation also shows explicitly the non-contiguous units. In the HAT representation in Figure 1a, for example, we can afford to remove the branches holding `don't` and `ne` and `pas` to obtain a HAT for two shorter sentences `I smoke` and `Je fume`, a translation unit not available as a phrase pair from this example. This should exemplify the utility of a recursive representation that is anchored in word positions.

2 Related Work

Most existing related work deals with factorizing permutations (bijective functions from a set of integers to itself) for efficient parsing of SCFGs [Zhang et al., 2006, Huang et al., 2009, Gildea et al., 2006]. Factorizing the right-hand sides of SCFG productions, where only nonterminals are aligned together bijectively but terminals are not aligned together, can be done efficiently in $O(n)$ [Zhang and Gildea, 2007b], improving over $O(n \log n)$ in [Gildea et al., 2006] and again over $O(n^2)$ in [Zhang et al., 2006]. The work in [Zhang and Gildea, 2007a] generalizes the algorithms above to higher dimensional permutations and provides combinatorial results regarding the factorization/decomposition of permutations. A special case of factorization is the binarization algorithms discussed in [Huang et al., 2009]. Like [Zhang et al., 2008], the present work is strongly related to factorizing arbitrary word alignments, a superset of the set of permutations, where the SCFG factorization algorithms belong.

Our work is also related to [Zhang et al., 2008] who present algorithms specifically meant for efficient extraction and compact representation of phrase pairs from word alignments. Zhang et al. extends existing work on efficiently computing the *common intervals*[2] of two permutations [Uno and Yagiura, 2000] and compactly representing all such common intervals [Landau et al., 2005, Bui-Xuan et al., 2005]. The goal of [Zhang et al., 2008] is to derive an optimal, linear-time algorithm for the problem of decomposing an arbitrary word alignment into SCFG rules such that *each rule has at least one aligned word* and is minimal in the sense that it cannot be further decomposed into smaller rules. The *normalized decomposition trees (NDTs)* of Zhang et al. 2008 [Zhang et al., 2008] represent compactly and efficiently all phrase pairs that the input word alignment induces. Figure 3 exhibits an example NDT from Zhang et al. 2008. The nodes of the NDT are labeled with pairs of phrase start-end positions, e.g., the node

[1]With some stretch of imagination we can view these operators as the starting point for compositionality operators akin to the traditional compositional machine translation [Landsbergen, 1982, Janssen, 1998].

[2]A common interval of two permutations is a shared set of integers that are consecutive in both permutations.

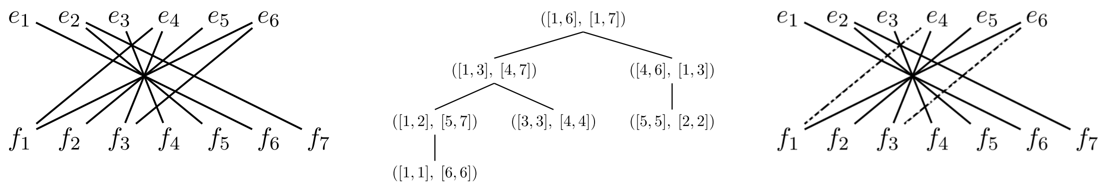

e_1 e_2 e_3 e_4 e_5 e_6
f_1 f_2 f_3 f_4 f_5 f_6 f_7
([1, 6], [1, 7])
([1, 3], [4, 7])
([4, 6], [1, 3])
([1, 2], [5, 7])
([3, 3], [4, 4])
([5, 5], [2, 2])
([1, 1], [6, 6])
e_1 e_2 e_3 e_4 e_5 e_6
f_1 f_2 f_3 f_4 f_5 f_6 f_7

Figure 3: (Left) an example word alignment and (Middle) its Normalized Decompostion Tree (NDT) (cf. Zhang, Gildea and Chiang 2008). (Right) word alignments with the same set of phrase pairs can be obtained by removing any of the dashed alignments links. The resulting word alignments have the same NDT shown in this figure.

labeled $([1,3],[4,7])$ covers the phrase pair with f-side $f_4 \dots f_7$ and e-side $e_1 \dots e_3$. This phrase pair breaks down into two children standing for adjacent, non-overlapping sub-phrase pairs: $([1,2],[5,7])$ and $([3,3],[4,4])$.

Our work differs from [Zhang et al., 2008] at various levels. We focus on the reordering aspects of word alignments and aim at making it explicit in node transduction operators in the HAT representation. Furthermore, we aim at representing every word alignment with a HAT that *uniquely* determines that word alignment. Because a phrase pair is a pair of strings without internal word alignment [Zhang et al., 2008], different word alignments that induce the same set of phrase pairs are represented by the same NDT. Figure 3 exhibits the example NDT from Zhang et al. 2008 with different word alignments that induce the same set of phrase pairs, i.e., the same NDT (see caption of Figure 3). For the sake of making the differences between HATs and NDTs explicit, Figure 4 exhibits a HAT for the word alignment to the left of Figure 3 and provides explanation regarding the HAT representation.

The empirical part of this paper (Section 6) explores the recursive reordering nature of word alignments using the HAT algorithm, providing a novel analysis where word alignments in existing parallel corpora are bucketed according to their recursive reordering properties as represented by their HATs. Using this analysis, we contribute to an ongoing debate regarding the empirical stability of the ITG hypothesis: the ITG hypothesis states that all (or at least the vast majority of the correct) word alignments (in any parallel corpus) can be represented as ITG STPs: for every sequence of sister nodes, their links can only be in one of two possible orientations, either fully monotone or fully inverted. The stability of the ITG assumption is an empirical matter that depends on the relative frequency of complex permutations and alignment constructions (like Discontinuous Translation Units [Søgaard and Kuhn, 2009]) in actual data. Our contribution to the ITG-coverage debate is distinct from earlier approaches in that it formally builds the HATs for *every* word alignment before doing any measurements: the measurements are stated in terms of formal properties of the produced HATs.

3 Defining Translation Equivalence over Alignments

Throughout this paper we will be concerned with *sentences*, finite sequences of tokens (atomic or terminal symbols). For the purposes of intuitive and simple exposition we will often talk about words but the treatment will apply to atomic tokens at other granularity levels.

Definition 1 (Alignment and sub-alignment)**.** *Given a source and target sentence pair,* $\mathbf{s} = s_1, \dots, s_n$ *and* $\mathbf{t} = t_1, \dots, t_m$*, we define an alignment* $\mathbf{a}$ *as a relation of pairs consisting of a position in* $\mathbf{s}$ *and another in* $\mathbf{t}$ *or NULL, i.e.,* $\mathbf{a} \subseteq \{0, 1, \dots, n\} \times \{0, 1, \dots, m\}$ *where position* 0 *stands for NULL. Each individual pair is a* link*. We will call* $\mathbf{b}$ *a sub-alignment of an alignment* $\mathbf{a}$ *when* $\mathbf{b} \subseteq \mathbf{a}$*.*

Next we provide a general definition of relations of translation equivalence.

Definition 2 (Translation-admissable sub-alignments)**.** *Given an alignment* $\mathbf{a}$ *between* $\mathbf{s}$ *and* $\mathbf{t}$*, a non-empty sub-alignment* $\mathbf{b} \subseteq \mathbf{a}$ *is* translation-admissable (t-admissable) *iff for every* $\langle x, y \rangle \in \mathbf{b}$ *holds*

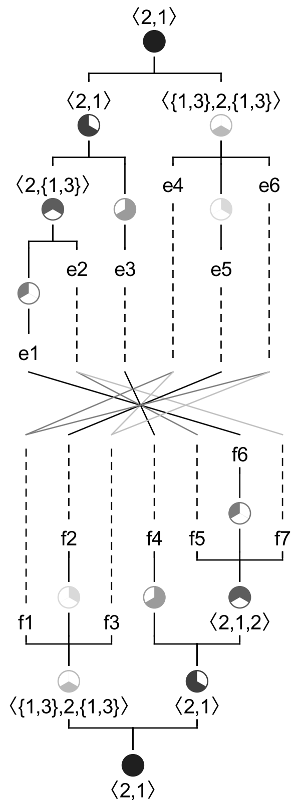


Figure 4: A visualization of a HAT for the word alignment to left of Figure 3. For convenience, here we show both the source and target side HATs linked together, i.e., a dual source-target HAT with node-links. Like in the case of NDTs, there could be various HATs for a given word alignment, of which one is chosen canonically to represent the rest. Nodes are depicted as circles with a filling: nodes with the same filling are linked together. For example, the node dominating $e_1 e_2$ is linked with the node dominating $f_5 \ldots f_7$. The operator above the e-side node $\langle 2, \{1, 3\}\rangle$ tells us that the first child of this e-node is linked with 2^{nd} child of the f-side node and that the second child of this e-node is linked with both the 1^{st} and 3^{rd} f-side children. Furthermore, the fact that e_2 is not dominated separately by a node (unlike e_1) represents the fact that e_2 is not on its own the e-side of a phrase pair because it is linked with the discontinous f_1 and f_3.

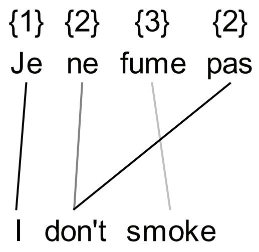


Figure 5: A word aligned sentence pair.

$\{\langle x1, y1\rangle \in \mathbf{a} \mid (x1 = x) \vee (y1 = y)\} \subseteq \mathbf{b}$. *In other words, all alignments involving source position x in* $\mathbf{a}$ *must either all be in* $\mathbf{b}$ *or else none, and similarly for all alignments involving target position y.*

In Figure 5, since `ne` and `pas` are both linked with `don't`, it is reasonable to think that `don't` translates as `ne + pas`. Hence the definition of t-admissable sub-alignments. For computational and representational reasons, we will be interested in a subspace of the t-admissable sub-alignments, particularly the *word aligned* phrase pairs reminiscent of phrase-based translation, e.g., [Zens et al., 2002, Koehn et al., 2003]. Intuitively, for standard phrase pairs, links are grouped together into larger units of translation equivalence if they are adjacent both at the source and target sides.

Definition 3 (Phrase-pair sub-alignment)**.** *A t-admissable sub-alignment* $\mathbf{b} \subseteq \mathbf{a}$ *is called a* phrase pair sub-alignment *of* $\mathbf{a}$ *iff both the sets of source and target positions in* $\mathbf{b}$ *(minus the NULLs – position zero), constitute contiguous sequences of source and target positions in* $\mathbf{a}$.

Definition 4 (Minimal phrase-pair sub-alignment)**.** *A phrase pair sub-alignment is minimal if and only if none of its proper sub-alignments is a phrase pair.*

In Figure 5, the sub-alignment representing the linking of $\{\langle \texttt{Je}, \texttt{I}\rangle, \langle \texttt{fume}, \texttt{smoke}\rangle\}$ is t-admissable for this alignment, whilst it is not a phrase pair sub-alignment.

Phrase pairs vs. phrase pair sub-alignments In contrast with phrase pair sub-alignments, phrase pairs (as defined in MT (and used, e.g., in [Zhang et al., 2008]) conflate the differences between word alignments because they do not retain the original word alignments. Figure 6 exhibits (schematic sketches of) three word alignments that constitute minimal phrase pairs of the same string pair, yet they constitute three different phrase pair sub-alignments. It is easy to see that every word alignment induces a **unique set** of phrase pair sub-alignments (henceforth **uniqueness** property). This subtle difference is crucial for representing word alignments uniquely as HATs.

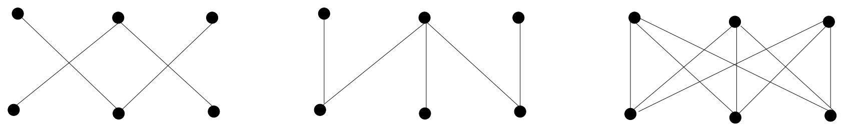


Figure 6: Three word alignments that constitute minimal phrase pairs

Unaligned words (NULLs) lead to an extensive notational burden and, in principle, in the sequel we will not provide a formal treatment of unaligned words, but our intuitive suggestions for this special case will aim at showing that an extension of the present techniques to unaligned words is easy to obtain.

4 Alignments as *S-permutations*: Relative Word Order

In this section we introduce *s-permutations (for set permutations)*, an asymmetric representation of word alignments that generalizes the well-known permutations over integers. We show that for every word alignment there exists a unique, equivalent s-permutation and vice versa. S-permutations are attractive because they can be parsed into HATs in a monolingual algorithm that resembles parsing monolingual strings into parse trees, albeit without an explicit grammar. We start out from permutations as useful representations for a subclass of alignments (bijective) that naturally capture the simultaneous adjacency requirement of links on both sides that underlies phrase pair sub-alignments, as studied in, e.g., [Zhang et al., 2006, Gildea et al., 2006].

If $\mathbf{a}$ is bijective, then the positions on the one side can be described as a *permutation* of the positions on the other side. For example, if $\mathbf{a} = \{1\text{-}2, 2\text{-}1, 3\text{-}3, 4\text{-}4\}$ (with source positions coming first in the pairs), then the target positions constitute the permutation $\langle 2, 1, 3, 4\rangle$ relative to source positions.

Definition 5 (Permutations and shifted-permutations)**.** *A permutation π over the range of integers $[1..n]$ is a sequence of integers such that each integer in $[1..n]$ occurs* exactly once *in π.*

Definition 6 (Sub-permutation)**.** *A sub-permutation π_x of a permutation π is a contiguous subsequence of π.*

Sub-permutations clearly comply with the adjacency requirement of linked positions on both sides, which are required for phrase pair sub-alignments. The following rather straightforward lemma highlights the fact that bijective alignments and permutations can be used to define exactly the same sets of phrase pair sub-alignments (translation equivalence relations):

Lemma 1. *The set of phrase pair sub-alignments for a bijective alignment (permutation)* $\mathbf{a}$ *is equivalent to the set of sub-permutations of the permutation corresponding to* $\mathbf{a}$*.*

The proof of this lemma follows from the definitions of phrase pair sub-alignment for bijective permutations and the definition of sub-permutation.

Notation For traversing a permutation π from left to right we will employ indices to mark the current state of traversal: at start the *index* is zero and after moving one *position* to the right the index increments by one (i.e., index $j > 0$ stands between positions j and $j + 1$). The notation $\pi_{<j}$ and $\pi_{>j}$ refers to subsequences of π that are respectively the prefix ending with the integer at position j and the suffix starting with the integer at position $j + 1$. Note that these subsequences are not necessarily sub-permutations of π since they might consist of integers that do not define a range of successive integers. For example, in $\pi = \langle 2, 1, 3, 4\rangle$, we find sub-permutations $\pi_{<3} = \langle 2, 1, 3\rangle$ and $\pi_{>3} = \langle 4\rangle$, but $\pi_{>1} = \langle 1, 3, 4\rangle$ is not a sub-permutation.

We can also represent a permutation, e.g., $\langle 2, 1, 3, 4\rangle$, as ranging over singleton sets, i.e., $\langle\{2\}, \{1\}, \{3\}, \{4\}\rangle$. This allows us to encode non-bijective alignments, containing sub-alignments that are not 1:1, as extensions of permutations over sets of target positions relative to source positions. The sets of target positions imply grouping constraints defined by alignments that are not 1:1.

Back to our running example (Figure 5). If we take French as the source language then the representation (called s-permutation) of this alignment is $\langle\{1\}, \{2\}, \{3\}, \{2\}\rangle$, whereas if we take English as the source side the *s-permutation* should be $\langle\{1\}, \{2, 4\}, \{3\}\rangle$. To arrive at these s-permutations, simply scan the source (respectively target) side left to right word-by-word and for every word position note down in a set notation the target positions linked with that word. In the representation $\langle\{1\}, \{2\}, \{3\}, \{2\}\rangle$ (the French-side view) the set $\{2\}$ appears in the second and fourth positions signifying that the 2^{nd} English word is linked with both the second and fourth French positions. In other words, the two appearances of $\{2\}$ signify a grouping constraint for the second and fourth French positions with surrounding positions. This example should also highlight the *asymmetric* nature of this representation (just like permutations).

Now we define s-permutations by providing a recipe for obtaining them from alignments.

Definition 7 (S-permutations for non-bijective alignments)**.** *A* s-permutation *is a finite sequence of finite sets of integers such that the union of these sets constitutes a range of integers $[0..n]$.*

A s-permutation is obtained from an alignment $\mathbf{a}$ as follows: For every source position $i_s > 0$ (i.e., not NULL), we represent this position with a set $\mathbf{a}(i_s)$ that fulfills $j_t \in \mathbf{a}(i_s)$ iff i_s and j_t are linked together, and none is NULL, i.e., $j_t \in \mathbf{a}(i_s)$ iff $\langle i_s, j_t \rangle \in \mathbf{a}$ and $j_t \neq 0$.

If we view alignments asymmetrically (say from the source side), we find that every s-permutation represents a single alignment and that every alignment (viewed from the source side) can be represented by a single s-permutation.

Another example of a s-permutation is $\langle \{1,2\}, \{3\}, \{2,4\} \rangle$ which implies the alignment $\{1-1, 1-2, 2-3, 3-2, 3-4\}$ (where here we informally represent an alignment as a set of linked source-target pairs of positions $x - y$).

Definition 8 (Sub-permutation of a s-permutation)**.** *A sub-permutation of a s-permutation* $\pi = \langle s_1, \ldots, s_m \rangle$ *is a* contiguous *subsequence* $\langle s_i, \ldots s_j \rangle$ *(*$i \leq j$*) that fulfills the requirement that the union* $(s_i \cup \cdots \cup s_j)$ *of the sets* $s_i, \ldots s_j$ *constitutes a contiguous range of integers and for every integer* $x \in (s_i \cup \cdots \cup s_j)$ *holds* $x \notin (s_1, \cup \cdots \cup s_{i-1} \cup s_{j+1} \cup \cdots \cup s_m)$.

This definition demands contiguous chunks on both sides and that links from the same position (including discontinuous cases) must remain together. For example, for s-permutation $\langle \{1\}, \{2\}, \{3\}, \{2\} \rangle$ (Figure 5, French as source), the atomic subsequences $\langle \{1\} \rangle$ and $\langle \{3\} \rangle$ are sub-permutations. In contrast, the atomic subsequence $\langle \{2\} \rangle$ is not a sub-permutation because one copy of the 2 remains in the complement (the two copies together stand for a discontinuous alignment with English position 2). The subsequence $\langle \{2\}, \{3\}, \{2\} \rangle$ constitutes a sub-permutation, whereas $\langle \{1\}, \{2\} \rangle$ is not a sub-permutation because again one copy of the 2 remains in the complement.

Partial sets In a s-permutation, we refer to those sets that do not constitute an atomic sub-permutation (like $\{2\}$ in $\langle \{1\}, \{2\}, \{3\}, \{2\} \rangle$ or $\{2,5\}$ in $\langle 3, \{2,5\}, 1, 4 \rangle$) with the term *partial sets*. A partial set either shares positions with other partial sets or it consists of a non-singleton set that does not constitute a sub-permutation.[3]

In s-permutation $\langle \{1,2\}, \{3\}, \{2,4\} \rangle$ for example, we find that each of $\{1,2\}$ and $\{2,4\}$ separately are partial sets, whereas $\{3\}$ is an (atomic) sub-permutation.

Also under the latter definition we find that the set of phrase pair sub-alignments is equivalent to the set of sub-permutations as stated in the following lemma.

Lemma 2. *The set of phrase pair sub-alignments for an alignment (s-permutation)* $\mathbf{a}$ *is equivalent to the set of sub-permutations of the s-permutations corresponding to* $\mathbf{a}$*.*

Hence, the **uniqueness** property for sets of phrase pair sub-alignments carries over to sets of sub-permutations: every s-permutation induces a unique set of sub-permutations.

We will now define the following intuitively simple partial order relation over sub-permutations of the same permutation π:

Definition 9 (Partial order $<$ over s-permutations)**.** *Given a s-permutation* π_1 *over* $[i..j]$ *and another s-permutation* π_2 *over* $[k..l]$*. We will write* $\pi_1 < \pi_2$ *iff* $i \leq j < k \leq l$*. This relation extends naturally to sub-permutations also.*

In summary, s-permutations are asymmetric representations of alignments. In terms of translation equivalence relations, it is important to highlight the equivalence of the set of phrase pair sub-alignments defined by a given alignment to the set of sub-permutations of the corresponding s-permutation. This equivalence implies that we can represent alignments hierarchically if we succeed to represent s-permutations

[3]Partial sets are contiguous source-side sub-units in what is known as Discontinuous Translation Units (DTUs). These correspond to the two cases of a source side position aligned with a discontinuous set of positions on the target side or a target side position aligned with a discontinuous set of positions on the source side. [Søgaard and Kuhn, 2009].

hierarchically. As we shall see, because s-permutations are asymmetric they constitute a nice intermediate representation on the way from alignments to hierarchical representations.

5 The Hierarchical Structure of Sub-permutations: Recursive Translation Equivalence

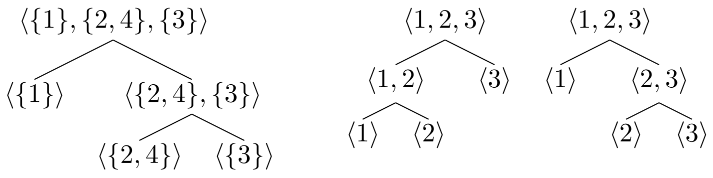


Figure 7: The left tree representation makes explicit the hierarchical structure of the sub-permutations of permutation set $\langle\{1\},\{2,4\},\{3\}\rangle$, and the two to the right for permutation $\langle 1,2,3\rangle$ (since all sets are singletons we simplified the s-permutation into a standard permutation). Note that there are two trees for the latter permutation, each showing the recursive grouping using different sub-permutations.

Consider the s-permutation $\langle\{1\},\{2,4\},\{3\}\rangle$ for the English side in Figure 5 (`I don't smoke`). In this s-permutation we see that the first English position is linked with first French position, and third to third, whereas the second English position (`don't`) is linked both with second and fourth French positions (`ne` and `pas`). The sub-permutations of this s-permutation are: $\langle\{1\}\rangle$, $\langle\{3\}\rangle$, $\langle\{2,4\},\{3\}\rangle$ and $\langle\{1\},\{2,4\},\{3\}\rangle$. The sequences $\langle\{2,4\}\rangle$ and $\langle\{1\},\{2,4\}\rangle$, for example, are not sub-permutations because $\{2,4\}$ does not constitute a range of consecutive integers.

Figure 7 (left side) shows how the sub-permutation $\langle\{1\},\{2,4\},\{3\}\rangle$ can be seen as the *concatenation* of the two sub-permutations $\langle\{1\}\rangle$ and $\langle\{2,4\},\{3\}\rangle$, and that the latter sub-permutation is the concatenation of $\langle\{2,4\}\rangle$ and $\langle\{3\}\rangle$, in this order. The same figure also shows two trees for the permutation $\langle 1,2,3\rangle$. Note how these two trees exhibit the grouping of different sub-permutations, that correspond to different sub-alignments, and hence also different translation equivalence relations.

In this section we are interested, on the one hand, in this kind of structuring of s-permutations into sub-permutations, and on the other, in a suitable representation that makes explicit the mapping between the local order of source-side groups and the order of their target-side counterparts. We define concatenation as the composition operation that we are going to assume here.

Definition 10 (Concatenation of sub-permutations and/or sequence of partial sets)**.** *The concatenation of sub-permutations is a special case of concatenation of sequences (ordered sets) because sub-permutations are sequences of sets of integers. The same applies to sequences of partial sets. The result of the concatenation of an ordered pair of sequences $\langle\pi_1,\pi_2\rangle$, written $concat(\pi_1,\pi_2)$ is the sequence of sets of integers obtained by concatenating the sequence π_2 after the sequence π_1. We define the concatenation operator to be left-associative.*

Note that concatenation itself is not guaranteed to lead to a sub-permutation even if both components are sub-permutations.

We will also define the segmentation of a (sub-)permutation (almost but not exactly the inverse of concatenation because concatenation is not guaranteed to result in a sub-permutation).

Definition 11 (Segmentation of a sub-permutation)**.** *A segmentation of a sub-permutation $\pi=\langle\sigma_1,\ldots,\sigma_n\rangle$ is a set of indices $B=\{j_0=0,j_1,\ldots,j_m=n\}$ that segments π into m adjacent, non-overlapping and contiguous subsequences (called* segments*) such that for all $0\leq i<m$ holds: the sub-sequence of π (segment) given by $\sigma_{j_i}^{j_{i+1}}=\sigma_{j_i}\ldots\sigma_{j_{i+1}}$ is either a* sub-permutation *of π or a sequence consisting of a* single partial set *from π.*

For example, the sub-permutation (showing the indices explicitly using extra subscript notation) $\pi_A = \langle_0\{1\},_1\{2\},_2\{3\},_3\{2\}_4\rangle$ has a possible segmentation $B_1 = \{0, 1, 4\}$ leading to sub-permutations $\langle\{1\}\rangle$ and $\langle\{2\}, \{3\}, \{2\}\rangle$, another segmentation $B_2 = \{0, 1, 2, 3, 4\}$ leading to a sub-permutations $\langle\{1\}\rangle$, $\langle\{3\}\rangle$ and twice the sequence with a single partial set $\langle\{2\}\rangle$. The set of indices $B^* = \{0, 2, 4\}$, for example, does not constitute a segmentation of π_A.

A second example might make the segmentations even clearer: for $\pi_B = \langle_0\{1\},_1\{2\},_2\{2\},_3\{3\}_4\rangle$ there are segmentations $B_1 = \{0, 1, 2, 3, 4\}$, $B_2 = \{0, 1, 3, 4\}$, $B_3 = \{0, 1, 4\}$ and $B_4 = \{0, 3, 4\}$ (note that $\langle\{2\}, \{2\}\rangle$ is a sub-permutation of π_B).

Segmentations and the hierarchical structure of sub-permutations The following lemma (with two sub-statements) is central for devising an algorithm for the hierarchical representation of sub-permutations in s-permutations. Intuitively, the two sub-statements in this lemma together imply that we can build recursive tree hierarchies that work with *minimal segmentations* of π and still cover *all sub-permutations*.[4]

Lemma 3 (Sub-permutations and minimal segmentations)**.** *The lemma has two subsections:*

Seg1 For every sub-permutation π_x of another sub-permutation π there exists a segmentation of π into a sequence of segments $\langle A_1, \ldots, A_m\rangle$ in which π_x is a member, i.e., there exists $1 \leq i \leq m$ such that $A_i = \pi_x$.

Seg2 Let $k > 1$ be the minimal cardinality of a segmentation of a sub-permutation π. We will refer to B with $|B| = k$ as a minimal segmentation. For every segmentation B of π, there exists a segmentation B_{min} of π such that $B_{min} \subseteq B$ and $|B_{min}| = k$. In other words, every non-minimal segmentation B can be regrouped into (is reachable from) a minimal segmentation.

Proof. **Seg1** By contradiction. Let us assume there is no such segmentation of π. If $\pi_x \neq \pi$ is a sub-permutation of π found between positions indexed with i and j, then there exists X_l and X_r, at least one of which is non-empty, such that $\pi = \langle X_l,\ _i\,\pi_x,\ _j\,X_r\rangle$ (the subscripts $_i$ and $_j$ are used to mark the indices). If π_x is not a member in any segmentation of π, then (by Definition 11) for every segmentation B of π holds $\{i, j\} \not\subset B$. Because the sub-sequence between i and j is a sub-permutation π_x, this implies that either one or both of the sub-sequences X_l and X_r cannot be segmented into sub-permutations and single partial sets. But because $Concat(X_l, \pi_x, X_r) = \pi$ is a sub-permutation, it is a sequence of sets over a range of consecutive integers $[n_l..n_i..n_j..n_r]$, where π_x is defined over the *proper sub-range* $[n_i..n_j]$. Hence, the integer sets in X_l and X_r must be defined as subsets of $[n_l..n_{i-1}n_{j+1}..n_r]$. But this implies that each such integer set in itself is either a partial set or can form on its own a sub-permutation of π. Contradiction, because this does constitute a segmentation B of π such that $\{i, j\} \in B$.

Seg2 Let π be a sub-permutation with a minimal segmentation of cardinality $|B_{min}| = k$. For every segmentation B of π we want to prove that there exists a segmentation of π called $B^{'}$ such that $B^{'} \subseteq B$ and $|B^{'}| = k$. The proof is by induction on $m = (|B| - k)$. For the case $m = 1$: By contradiction. Suppose there is a segmentation B of π with $|B| = k+1$ for which there exists no minimal segmentation $B^{'} \subset B$ and $|B^{'}| = k$. Because B segments π into a sequence $X_1, \ldots, X_{k+1}$ of $k+1$ sub-permutations or partial sets, the assumption holds iff *for all consecutive pairs* X_i and X_{i+1} ($1 \leq i \leq k$) holds: $\langle X_i, X_{i+1}\rangle$ does not constitute a sub-permutation. This situation can occur iff for all $1 \leq i \leq k$ holds the intersection of X_i and/or X_{i+1} with $\cup_{j \notin \{i,i+1\}} X_j$ is non-empty (multiple discontiguous source-side positions aligned with the same target position) and/or, the converse, the set $X_i \cup X_{i+1}$ does not constitute a consecutive range of integers. But if this holds for $\underline{\text{all consecutive pairs of segments}}$ in B, then by the definition of segmentation in every consecutive pair $\overline{\text{of segments at least one segment}}$ is a partial set. In that case, B constitutes the only possible segmentation of π of length $k+1$ because at least every other segment in B is a partial set. Because segmentation works with consecutive segments and all consecutive

[4]Minimal segmentations underlie SCFG rule factorization algorithms, e.g., [Zhang et al., 2006, Gildea et al., 2006]. The main difference is that SCFG rules are permutations whereas here we are dealing with a generalization into s-permutations. The algorithm for devising NDTs in [Zhang et al., 2008] employs a similar notion where phrases are segmented into sub-phrases. However, our segmentation of s-permutations corresponds more explicitly with SCFG rule factorization.

pairs in B cannot form a sub-permutation, there is no way to segment π into k segments. This contradicts our assumption that π has a minimal segmentation of length $k > 1$. This concludes the case $m = 1$ in the induction.

For segmentations B with $m = (|B| - k) > 1$ we will exploit the standard induction assumption: for every segmentation B_x of π which fulfills $1 \leq |B_x| - k < m$ there exists a segmentation of π called $B^{'}$ such that $B^{'} \subset B_x$ and $|B^{'}| = k$. We want to prove that the same holds for every segmentation $|B| = m + k$. By contradiction, let us assume there exists $|B| = m + k$ for which there does not exist a $B^{'} \subset B$ and $|B^{'}| = k$. This can be true iff we cannot reduce B in any way to a segmentation $B_1 \subset B$ such that $k \leq |B_1| < (m + k)$. For if we could find such a segmentation B_1, the induction assumption will apply; in other words, B can be reduced into B_1 first, and by the induction assumption there exists $B^{'} \subset B_1 \subset B$ and $|B^{'}| = k$. But if there exists no segmentation $B_1 \subset B$ such that $k \leq |B_1| < (m + k)$ then *none* of the sequences x of adjacent segments in B of length $2 \leq |x| \leq (m + 1)$ can be reduced into a sub-permutation. Moreover, none of the adjacent segments of length $|x| > (m + 1)$ can be reduced into a sub-permutation because that would contradict the main assumption and the minimality of k for π. But that implies that B segments π in such a way that none of the sequences of segments in B of length $2 \leq |x| < |\pi|$ can be reduced into a sub-permutation. The latter implies that every sequence of length $2 \leq |x| < |\pi|$ must consist of at least one partial set that has its complement outside the sequence in π. But that implies that B and thus also π is a sequence of partial sets such that each partial set has its complements at $|\pi|$ segments away from its location, which can be true iff π cannot be segmented at all into a segmentation of length larger than one. This contradicts the existence of a minimal segmentation of π of length $k > 1$. $\square$

These two lemma's together prove that every sub-permutation π_x of a sub-permutation π can be a segment in a segmentation B of π which is a superset of a minimal segmentation of π.

Next we take a little detour to define Synchronized Tree Pairs (STPs): this is a *rather constrained* node linking relation between tree pairs (akin to ITGs [Wu, 1997]). The explicit definition of STPs aims at highlighting the bijective and "vertically non-crossing" node linking relation. After defining STPs we return to our main task of representing the structure of s-permutations as hierarchical structures of their sub-permutations and the reordering they imply.

5.1 Synchronized Tree Pairs with Layered Linking

Because the labels in the trees assumed here are irrelevant to the discussion, we may assume that internal nodes in these trees are unlabeled (or labeled with a single symbol called *bracket*) and that leaf nodes are labeled with integers representing positions in sentence pairs (the terminals).

Definition 12 (Synchronized Tree Pairs (STPs))**.** *A Synchronized Tree Pair (STP) $\langle \tau_s, \tau_t, \sim \rangle$ consists of a pair of trees τ_s and τ_t and a node-linking relation $\sim$ which at least links the roots of both trees. Apart from the roots, any other node in τ_s could possibly (but not necessarily) be linked together with nodes in τ_t.*

Wu [Wu, 1997] defines BITGs that derive binary STPs $\{\langle \tau_s, \tau_t \rangle\}$ each fulfilling strict criteria on the node-linking relation. Graphically speaking, links do not cross "vertically" but may cross "horizontally". We state this as the *layered linking property*, also defined by Wu [Wu, 2010] as the *no-crossing constraint*.

Definition 13 (Layered linking)**.** *The linking relation $\sim$ between the nodes of τ_s and τ_t is a* correspondence *(left-total and right-total*[5]*), and for every pair of nodes μ_s in τ_s and μ_t in τ_t holds: if $\langle \mu_s, \mu_t \rangle \in \sim$ then the children of μ_s are linked only with the children of μ_t and vice versa.*

[5]In simple words, every node in τ_s is linked with some node in τ_t and vice versa.

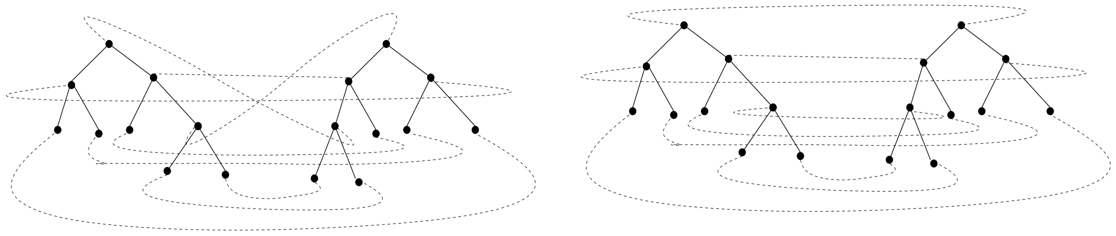


Figure 8: In the left-hand STP the roots are involved in vertically crossing links leading tot non-layered links. In the right-hand, a layered linking STP is exemplified.

Figure 8 shows a schematic example of one layered linking and another non-layered linking (in this abstract figure the links are visualized as edges).

For s-permutations, the attractive aspects of STP's with layered node linking is that they can be represented as a single source tree decorated at every internal node μ_s with a local transducer that explains how the child-order of the linked target node μ_t is obtained as an order permutation of the child-order of μ_s, plus or minus a handful extra operations for dealing with many-to-one and one-to-many alignments, i.e., s-permutations.

5.2 Representing s-permutations with STPs: Requirements

The algorithms for structuring permutations and s-permutations into layered-linking STPs, presented respectively in Subsections 5.3 (and 5.4), take as input a permutation (and s-permutation respectively) π and output a finite set of STPs. These algorithms abide by three requirements stated in terms of the correspondence between sub-permutations of an s-permutation and linked nodes in the STPs built by these algorithms.

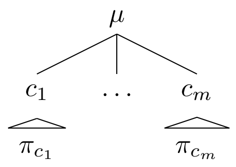


Figure 9: A schematic view of decomposing a sub-permutation π into sub-permutations $\pi_{c_1}, \cdots, \pi_{c_m}$.

I1 (Soundness) For every linked node in every STP representing input s-permutation π, the fringe dominated by that node corresponds to a sub-permutation of π. We will say that the linked node dominates that sub-permutation and that the latter is dominated by the node.

I2 (Completeness) For every sub-permutation π_x of input s-permutation π there will be a linked node that dominates π_x at least in one of the STPs that the algorithm generates for π.

I3 (Maximal hierarchy) Let linked node μ have a sequence of $m > 1$ child nodes $c_1, \dots c_m$, and denote with $\pi_{c_1}, \dots \pi_{c_m}$ the sequence of fringes (sub-permutations or partial sets) dominated by $c_1, \dots c_m$ (See Figure 9). Two statements hold: (A) The sub-permutation π_μ dominated by node μ is equivalent to $concat(\pi_{c_1}, \dots \pi_{c_m})$, and (B) m is the cardinality of a minimal segmentation of π_μ.

Requirements (I1) and (I2) together guarantee that there is a one-to-one mapping between the linked nodes in the STPs built by the algorithm and the sub-permutations of the input s-permutation. Requirement (I3) guarantees that the tree structure is meaningful and maximal as a hierarchical organization of sub-permutations. The tree structure is compact in that it organizes sub-permutations that concatenate together into larger sub-permutations resulting from the concatenation, and it is maximal because when $m > 1$ is the minimal segmentation of a sub-permutation into sub-permutations, the tree structure will be as deep as possible and contain as many linked nodes as possible (given all other requirements on what the structure is supposed to represent). Figure 9 shows example trees for structuring the permutation $\langle 1, 2, 3, 4\rangle$. The nodes labeled with x do not abide by the minimal segmentation requirement, whereas nodes that are unlabeled do so.

It is crucial here to stress the fact that the requirements (I1-3) are expressed in terms of sub-permutations of the s-permutation.[6] Hence, the soundness and completeness requirements (I1 and I2) together guarantee that the nodes and the linking relation in a generated STP exactly represent the set of sub-permutations of the input s-permutation. In this sense, we can now state explicitly the following important property:

I4 (Uniqueness) If the algorithms generate an STP τ for a s-permutation π, then there does not exist a s-permutation $\pi_2 \neq \pi$ for which τ will also be generated by these the algorithms.

The uniqueness property (I4) follows from I1, I2 and the uniqueness property for sets of sub-permutations. This uniqueness property is one aspect where the representation introduced in this work differs from existing representations, particularly NDTs [Zhang et al., 2008]. Further differences will arise in the next section when every STP will be represented by a single tree with nodes decorated by s-permutations (which are interpreted as transduction operators).

5.3 Permutation Trees: Hierarchies over Sub-permutations of Permutations

It is instructive to start with the known case of factorizing permutations before extending it to s-

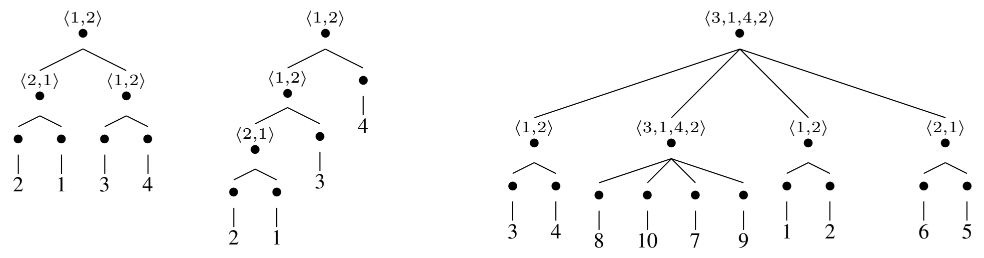


Figure 10: Permutation trees: left two for the same permutation, right the single PeT for the permutation. Note that the identity operators are not marked on pre-terminal nodes but left implicit.

permutations. Conceptually, our factorization algorithm for permutations share the same basic intuition as the more efficient algorithms that compute a single factorization [Zhang et al., 2006, Gildea et al., 2006].

A Permutation Tree (see also [Gildea et al., 2006]) for a given permutation π over $[1..n]$ is a representation equivalent to layered-linking STPs consisting of a tree representation over sub-permutations of π and node-links represented as local *permutation operators* decorating every node; a permutation operator on node μ stipulates how to permute the ordered sequence of the children of μ to obtain the sequence of children of the node that μ is linked with. Permutation Trees (PeTs) extend the kind of trees generated by ITGs because they allow arbitrary layered linking relations as opposed to the inverted/monotone binary choice that ITG strictly postulates. For every permutation we will define a unique set of PeTs that fulfills the requirements (I1-I3) from Section 5.2.

Figure 11 shows Algorithm 1 for building the set of PeTs for an input permutation. In a CYK-like fashion [Younger, 1967], Algorithm 1 employs a data structure (chart) consisting of entries $\mathbf{Entry}(i, j)$, for pairs of integers $i < j$ in π. The algorithm's intended invariant property is that $\mathbf{Entry}(i, j)$ stands for the set of all root-nodes of derivations that cover the sub-permutations between two non-negative integers $i < j$. A root-node will be represented as a unique address τ and a vector of $0 \leq m$ child nodes

[6] Recall that sub-permutations, unlike mere phrase pairs, retain the alignments internally.

ALGORITHM 1 ($PeTs(\pi)$). **Input:** *A (sub-)permutation* $\pi = k_1, \cdots k_n$ *over* $[(l+1)..(l+n)]$ *for some* $l \geq 0$.
Output: *Returns* $\mathbf{Entry}(\pi)$*, the set of root nodes of Permutation Trees for* π.

```
BEGIN
 If $\mathbf{Entry}(\pi) \neq \emptyset$ then Return $\mathbf{Entry}(\pi)$;
 else
     For every minimal segmentation $B$ of $\pi$ into $m \geq 1$ sub-permutations $\pi_1, \cdots, \pi_m$ do

    Case $m > 1$ # π segments into more than one sub-permutation
         1  For each $\pi_i \in \{\pi_1, \cdots, \pi_m\}$ do $\mathbf{Entry}(\pi_i) := PeTs(\pi_i)$;
         2  For each $\langle\tau_1, \cdots, \tau_m\rangle \in (\mathbf{Entry}(\pi_1) \times \cdots \times \mathbf{Entry}(\pi_m))$ do
            • Allocate a fresh node address $\tau$;
              $\mathbf{Entry}(\pi) := \{(\tau, \langle\tau_1, \cdots, \tau_m\rangle)\} \cup \mathbf{Entry}(\pi)$;
            • Label $\tau$ with a permutation operator $O = \langle o_1, \cdots, o_m\rangle$ over $[1..m]$
              such that for every two positions $1 \leq i \neq j \leq m$ holds: $\pi_{o_i} < \pi_{o_j}$ iff
              $o_i < o_j$.


    Case $m == 1$ # m==n==1, i.e, a trivial singleton permutation
         1  Allocate a fresh node address $\tau$;
            Label $\tau$ with the identity operator;
            Allocate a fresh leaf node address $\tau_l$ and label it $\max(\pi)$;
            $\mathbf{Entry}(\pi) := \{(\tau, \langle\tau_l\rangle)\}$;

   Return $\mathbf{Entry}(\pi)$;

END
```

Figure 11: Algorithm $PeTs(\pi)$ outputs the set of PeTs for input permutation π.

$\langle\tau_1, \cdots, \tau_m\rangle$. Initially $\mathbf{Entry}(i, j) := \emptyset$ for all valid pairs i and j. As a shorthand, we use the notation $\mathbf{Entry}(\pi)$ to stand for $\mathbf{Entry}(\min(\pi) - 1, \max(\pi))$, i.e., covering the "span" between from i to j.[7]

In Figure 11, when π is non-trivial (i.e., $n > 1$), the algorithm builds the PeTs recursively each time by segmenting π into a minimal sequence of sub-permutations and then building sets of PeTs recursively for these sub-permutations. The PeTs for π are obtained by combining PeTs from these sets and putting them under a single node dominating the segmentation. Like standard CYK, the worst case complexity is $O(n^3)$. See also [Gildea et al., 2006] for an $O(n)$ implementation building a single PeT per permutation rather than a packed forest. The latter can be achieved by replacing in Algorithm 1 the loop "For every minimal segmentation" with "For some canonically chosen minimal segmentation" and at **2** in $m > 1$ the "For each $\langle\tau_1, \cdots, \tau_m\rangle$" with a canonical choice of of a single vector of children.

Figure 10 exhibits the two PeTs for permutation $\langle 1, 2, 3, 4\rangle$, and the single PeT for permutation $\langle 3, 4, 8, 10, 7, 9, 1, 2, 6, 5\rangle$. The latter case is interesting because the minimal segmentation of this permutation consists of four sub-permutations.

Theorem 1 (Soundness, completeness and maximal hierarchy of $PeTs(\pi)$)**.** *The set of PeTs output by Algorithm* $PeTs(\pi)$ *fulfills the requirements (I1, Soundness), (I2, Completeness) and (I3, Maximal hierarchy) stated in detail in Section 5.2.*

Proof. **Soundness** By construction: every node created by $PeTs(\pi)$ is built for a sub-permutation. This

[7]For a permutation π, the notation $\min(\pi)$ and $\max(\pi)$ stands respectively for the minimum and maximum integers in π.

is easy to check for the case $n = 1$ as well as for $n > 1$.

Completeness The proof hinges on Lemma 3. We will proceed by induction on the length of π. The cases $n = 1$ and $n = 2$ are trivial and are easy to prove as base for induction. We concentrate on the induction step to length $n > 2$, where we will use the induction assumption for all (sub-)permutations shorter than n. The proof of the induction step is by contradiction. Suppose now there is a sub-permutation π_x of π that is not dominated by any node in a PeT built by $PeTs(\pi)$. Necessarily $\pi_x \neq \pi$ and $n > 1$ because the algorithm must build a root node for every input permutation π. There exists a segmentation B of π in which π_x participates (by Lemma 3, part Seg1). Because π_x is assumed not to have a node in any PeT, B cannot be minimal (because if it were minimal then Algorithm $PeTs(\pi)$ would build a node for it). However, if B itself is not minimal, then there is a hierarchical structuring (tree) of B (by concatenation of sub-permutations into new sub-permutations) leading to a minimal segmentation B_{min} of π (by Lemma 3, part Seg2). Consequently, π_x will be covered within one of the sub-permutations called $\pi_{(x)}$ of B_{min}. By the induction assumption and because $\pi_{(x)}$ must be strictly shorter than π, we find that π_x will have a node built for it within the PeTs for $\pi_{(x)}$. Because B_{min} is a minimal segmentation of π, Algorithm $PeTs(\pi)$ will build a root node dominating a child for every sub-permutation in the sequence defined by B_{min}, i.e. also for $\pi_{(x)}$. Particularly, a copy of the node built for $\pi_{(x)}$ will constitute the root for every PeT in $PeTs(\pi_{(x)})$. Contradiction because now there is a node corresponding for π_x in a PeT built for π.

Maximal hierarchy This follows from the explicit minimal segmentation points in the algorithm, Lemma 3 and the preceding proof.

$\square$

Binary Inversion-Transduction Trees (BITTs) Consider for example the permutation $\langle 2, 1, 3, 4 \rangle$: both PeTs for this permutation are fully binary branching (see Figure 10). A well-known example of a permutation that does not have binary branching PeTs is $\langle 2, 4, 1, 3 \rangle$ [Wu, 1997]. The latter permutation is called a *non-binarizable permutation*. Permutations that have fully binary branching PeTs are called *binarizable* permutations [Huang et al., 2009].

5.4 Hierarchical Alignment Trees (HATs) for s-permutations

Figure 12 shows Algorithm 2 which builds the set $HATs(\pi)$ for any s-permutation π. It generalizes Algorithm 1 and the differences are local. We will refer to the STPs built by Algorithm 2 with the name *Hierarchical Alignment Trees* (HATs), in order to distinguish them from PeTs and BITTs, which are simpler versions of HATs. One notational difference is in the definition of $\min(\pi)$ and $\max(\pi)$ for a s-permutation π: the minimum and maximum in this case are over the union of the sets of integers in π.

Observe in Algorithm 2 an important difference with Algorithm 1 for the case ($m == 1$): Algorithm 2 discriminates between partial sets (which are represented as leaf nodes) and sub-permutations, which are built as pre-terminal nodes dominating a leaf node. The importance of making this distinction can be understood from the fact that all members of a partial set must remain together (under the same mother node) because none of them separately corresponds to a target-side node. This is in contrast with sub-permutations.

Another difference is that Algorithm 2 defines the node operators as s-permutations. For step **2** in Figure 12, the s-permutation is initially built with "place holders" (step **a**) that are normalized (step **b**) to get rid of gaps. For example, in the s-permutation $\langle 3, \{2, 6\}, 1, 4, 5 \rangle$ (see Figure 16) the algorithm will reduce sub-permutation $\langle 4, 5 \rangle$ into a single position (step **a**), leading to $\langle 3, \{2, 6\}, 1, 4 \rangle$ which does not constitute a s-permutation (not a contiguous range of integers). In (step **b** function $MakePermS()$ in Figure 13), these "place holders" are reduced to a consecutive range of integers: the 6 in $\langle 3, \{2, 6\}, 1, 4 \rangle$ is exchanged with 5, leading to $\langle 3, \{2, 5\}, 1, 4 \rangle$. See also the caption of Figure 13.

Figure 14 contains the running `ne ⋆ pas` example represented as HAT and unpacked into an STP.

ALGORITHM 2 ($HATs(\pi)$). **Input:** *A sub-permutation / s-permutation* $\pi = \langle \mathbf{k}_1, \cdots \mathbf{k}_n \rangle$*, where each* $\mathbf{k}_i$ *is an integer set.*
Output: $\mathbf{Entry}(\pi)$ *is the set of HATs for* π*.*

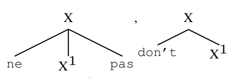
```
BEGIN
  If $\mathbf{Entry}(\pi) \neq \emptyset$ then Return $\mathbf{Entry}(\pi)$;
  else
     For every minimal set of indices $B = \{j_0 = 0, j_1, \cdots, j_m = n\}$ segmenting $\pi$ into $m \geq 1$
     segments such that for all $0 \leq i < m$, $\pi_{i+1} = \langle \mathbf{k}_{j_i} \cdots \mathbf{k}_{j_{i+1}} \rangle$ do

       Case $m == 1$ # m==n==1
            if ($\pi$ is a single partial set)
            then Allocate a unique leaf node $\tau$ and label it $\pi$;
                 $\mathbf{Entry}(\pi) := \{(\tau, \langle\rangle)\}$;
            else # $\pi$ must be a sub-permutation
                 • Allocate a unique node $\tau$ and a leaf node $\tau_l$ labeled $\pi$;
                 • Label root $\tau$ with the identity operator;
                 • $\mathbf{Entry}(\pi) := \{(\tau, \langle \tau_l \rangle)\}$;

       Case $m > 1$
            1  For each $\pi_i \in \{\pi_1, \cdots, \pi_m\}$ do  $\mathbf{Entry}(\pi_i) := HATs(\pi_i)$;
            2  For each $\langle \tau_1, \cdots, \tau_m \rangle \in (\mathbf{Entry}(\pi_1) \times \cdots \times \mathbf{Entry}(\pi_m))$ do
               •  Allocate a unique node address $\tau$;
                  $\mathbf{Entry}(\pi) := \{(\tau, \langle \tau_1, \cdots, \tau_m \rangle)\} \cup \mathbf{Entry}(\pi)$;
               •  Label $\tau$ with a s-permutation $\mathbf{O} = \langle \mathbf{o}_1, \cdots, \mathbf{o}_m \rangle$ built as follows:
                  a  For $0 \leq i < m$ do  # build $m$ integer sets
                     if ($\pi_{i+1}$ is a sub-permutation) then $\mathbf{o}_{i+1} = \min \cup_{x=j_i}^{j_{i+1}} \mathbf{k}_x$;
                     else $\mathbf{o}_{i+1} = \pi_{i+1}$; # for partial sets
                  b  # The set $\cup_{i=1}^{m} \mathbf{o}_i$ might not constitute a proper integer range
                     # $MakePermS()$ (see Figure 13) corrects this.
                     $\mathbf{O} := MakePermS(\mathbf{O})$;

     Return $\mathbf{Entry}(\pi)$;
END
```

Figure 12: Algorithm $HATs(\pi)$ outputs the set of HATs for input s-permutation π.

Nodes that are represented as ● dominate sub-permutation of the s-permutation, and they are linked nodes on both sides of the STP. Particularly note how a node dominates `fume` (inside the `ne fume pas`) signifying that it constitutes a sub-permutation represented by ● and linked with a node dominating `smoke`. This does not hold for the words `ne`, `pas` and `don't`. They are grouped together only including `fume` and `smoke`. Note also that we could extract pairs of linked subtrees from this STP only at the nodes marked with ● and thereby extract all sub-permutations, i.e., phrase pairs. If we would allow the subtrees to end at frontier nodes marked ●, we would obtain also Chiang-like constructions representing synchronous productions like X→ ⟨ 

⟩ where X is a nonterminal variable of the synchronous grammar and the superscript X^1 stands for synchronized nonterminal leaf nodes (linked ● nodes in our STPs).

ALGORITHM 3 ($MakePermS(\mathbf{O})$). **Input:** $\mathbf{O} = \langle \mathbf{o}_1, \cdots, \mathbf{o}_m \rangle$, *such that* $\forall 1 \leq i \leq m : \mathbf{o}_i \subset \mathcal{N}$ *(the natural numbers);*
Output: $MakePermS(\mathbf{O})$ *is the s-permutation that* $\mathbf{O}$ *stands for;*

```
BEGIN

Let $size := | \cup_{i=1}^{m} \mathbf{o}_i |$;
# Create an monotonically ordered list $\mathbf{v}$ of the unique integers in $\mathbf{O}$
Let $\mathbf{v} = \langle \mathbf{v}[1], \cdots, \mathbf{v}[size] \rangle$ such that
    $(\forall 1 \leq j < size : \mathbf{v}[j] < \mathbf{v}[j+1])$ and $(\{\mathbf{v}[1], \cdots, \mathbf{v}[size]\} = \cup_{i=1}^{m} \mathbf{o}_i)$;
For $\mathbf{o}_i \in \{\mathbf{o}_1 \cdots, \mathbf{o}_m\}$ # Correct every set $\mathbf{o}_i$
  let $temp = \emptyset$;
  For all $k \in \mathbf{o}_i$ # Replace each integer $k$ with its index $x$ in $\mathbf{v}$.
    Find $x \in [1..size]$ such that $(\mathbf{v}[x] = k)$;
    $temp := temp \cup \{x\}$;
  $\mathbf{o}_i := temp$;

Return $\mathbf{O}$;
END
```

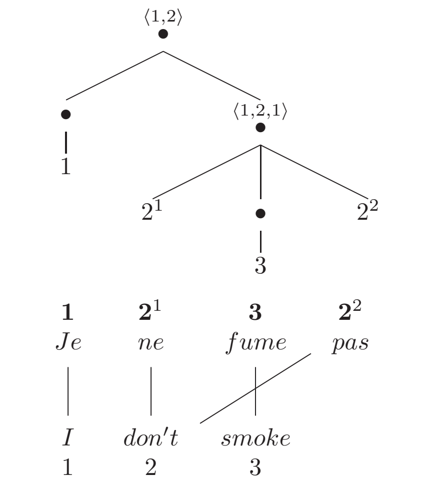
Figure 13: The union of sets in $\mathbf{O}$ might originally not constitute a range of integers (e.g., $\mathbf{O} = \langle \{1, 4, 8\}, \{5\}, \{2\} \rangle$). Algorithm $MakePermS(\mathbf{O})$ outputs the s-permutation for $\mathbf{O}$ (e.g., $\langle \{1, 3, 5\}, \{4\}, \{2\} \rangle$).

Like Algorithm 1 also Algorithm 2 fulfills the soundness, completeness and maximal hierarchy requirements, i.e., generalizing Theorem 1 from permutations to s-permutations. The proofs of this generalization are rather similar (but more detailed) and hinge on Lemma 3 (see Proof of Theorem 1). Importantly, Lemma 3 is stated and proven for *s-permutations in general*, which implies that it applies directly within the proof of the generalization of Theorem 1 to s-permutations (i.e., Algorithm 2). Given the central role of Lemma 3 within the proof, we skip the explicit statement of a corresponding theorem and proof.

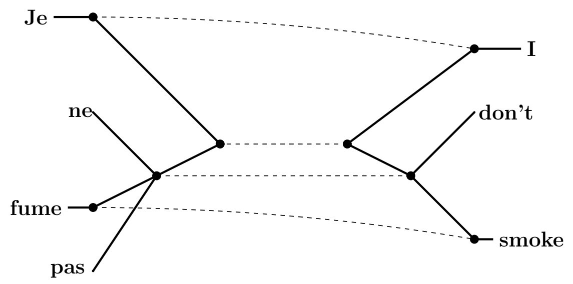
5.5 Efficiency and complexity issues

The complexity of implementing a chart-based version of Algorithm 2 depends mainly on the complexity of (1) identifying all sub-permutations of a given s-permutation and (2) on computing the minimal segmentations for every sub-permutation. Naively, identifying all sub-permutations and partial sets takes $O(n^2)$ time as it demands checking for every span $\langle i, j \rangle$ whether it is a sub-permutation. Following [Zhang et al., 2008], because sub-permutations are word aligned phrase pairs, their intersection, difference and union is also a sub-permutation. The algorithm of [Zhang et al., 2008] allows recognition of all sub-permutations in $O(n)$ time. This step can be done as preprocessing.

Having identified all sub-permutations and partial sets, the rest of the algorithm is very similar to standard CYK when a grammar is given, albeit here the grammar is *implicit in finding the minimal segmentations* of every sub-permutation. To identify the minimal segmentations of a sub-permutation of length m (there are $O(n^2)$ such sub-permutations), Dijkstra's shortest-path algorithm [Dijkstra, 1959] can be applied to the graph representing all eligible segmentations of the sub-permutation. The graph consists of m vertexes for a sub-permutation of length m and efficient versions of Dijkstra's algorithm take $O(m \log m)$ time. Hence, the worst-case complexity of Algorithm 2 is $O(n^3 \log n)$. Following [Zhang et al., 2008], an linear-time algorithm that computes a single canonically selected HAT for every word alignment can be obtained by simplifying two *for loops*: the loop concerning "for all segmentations" is replaced by selecting a single canonical segmentation, and the loop "for each $\langle \tau_1, \cdots, \tau_m \rangle$" by



(a) Bottom: A word aligned sentence pair. Top: The HAT representation.



(b) An STP representation showing the node-aligned pair of trees that the HAT in Figure 1a packs together.

1

Figure 14: In Figure 14a, the word-aligned sentence pair is shown with the French words decorated with the linked English positions (e.g., `Je` and `fume` are aligned respectively with English positions 1 and 3). We use extra superscript notation (as in 2^1 and 2^2) to represent the fact that French words `ne` and `pas` are first and second parts of a construction that aligns with English word in position 2 (`don't`). Figure 14a also shows the HAT representation. The notation $\langle 1, 2, 1\rangle$ on the node dominating the `ne fume pas` construction stands for a *transduction operator* which links the first and third children of this node (respectively dominating `ne` and `pas`) with the first child under the corresponding node on the English side; similarly the 2 in $\langle 1, 2, 1\rangle$ links the second child of this node with the second child of the corresponding node on the English side.

selecting a single tuple $\langle \tau_1, \cdots, \tau_m \rangle$.

As a side note regarding dealing with actual word alignments containing unaligned words, it is possible to first filter out the NULL aligned words leaving a NULL-free alignment, factorize the latter and then insert the NULL-aligned words into the chart where they were originally. The phrase pair semantics of translation equivalence dictates that every unaligned word must group with adjacent phrase pairs on either side. When the number of such unaligned words is large, this could lead to a very large number of s-permutations representing the same alignment. Crucially, the structure sharing[8] between the different s-permutations implies that this does not disturb the polynomial-time complexity of the algorithm. Nevertheless, the storage of the full HAT forest can be space-consuming in some extreme cases and for the experiments in Section 6 we will set a cut-off constant on the space, which rules out only a *negligible* number of alignments in the corpus.

5.6 The role of node operators

The semantics of a node operator that is a s-permutation π over the child sequence of the current node is a generalization of the semantics of the ITG operators [] and $\langle\rangle$. The semantics of these operators is the same as the definition of s-permutations for standard alignments but here it applies to the sequence of child nodes of the current node. Algorithmically speaking the semantics of the s-permutation π over the children of the current node μ linked with a target side node μ_t is obtained as follows. Scanning π left to right: for the set of integers X in the i^{th} position in π generate child positions under μ_t corresponding to the integers in X and link each of these target positions with the i^{th} child of μ.

Figure 15 exhibits three abstract word alignments that constitute minimal phrase pairs. In the HAT representation each of the three will be represented as a single linked node (pre-terminal level) dominating three terminal nodes. While the tree structure itself is the same, the node operators (s-permutations) on each are different. The node operators specify the internal alignment structure for, otherwise, hierarchically very difficult to represent word alignments. This way, the internal alignments of phrase pair units remain preserved.

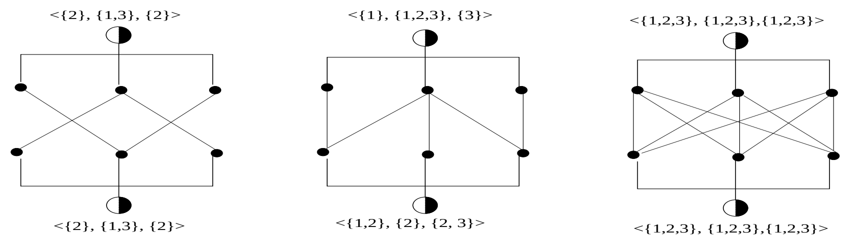


Figure 15: Three word alignments that constitute minimal phrase pairs and their HATs

Figure 16 exhibits an example word alignment (coming straight from our automatically aligned data) and a HAT on one side and its dual HAT on the other side, with links between the pairs of nodes (represented by a choice of circle filling). Note the discontinuous alignment of `owe` with `zijn` and `verschuldigd` together with another crossing alignment. The HAT representation *reveals* a pair of linked nodes dominating the synchronous pair $\langle (X_1 \texttt{ owe } X_2\ X_3),\ (X_2 \texttt{ zijn } X_1\ X_3 \texttt{ verschuldigd}) \rangle$, where X_1, X_2 and X_3 stands for three aligned pairs of nodes in the HAT. The latter bilingual construction is a Chiang-style production. The pair of linked nodes is decorated with a s-permutation $\langle 3, \{2, 5\}, 1, 4 \rangle$ on the English side and $\langle 3, 2, 1, 4, 2 \rangle$ on the Dutch side. Observe how these s-permutations actually link the second (`zijn`) and fifth (`verschuldigd`) children of the Dutch node with the second child (`owe`)

[8]In analogy to the monolingual case of parsing finite-state automata (lattices or word-graphs) with CFGs [van Noord, 1995, Sima'an, 1999], the bilingual case here also remains polynomial-time; the time complexity multiplies with a constant factor linear in the number of edges/transitions in the automaton.

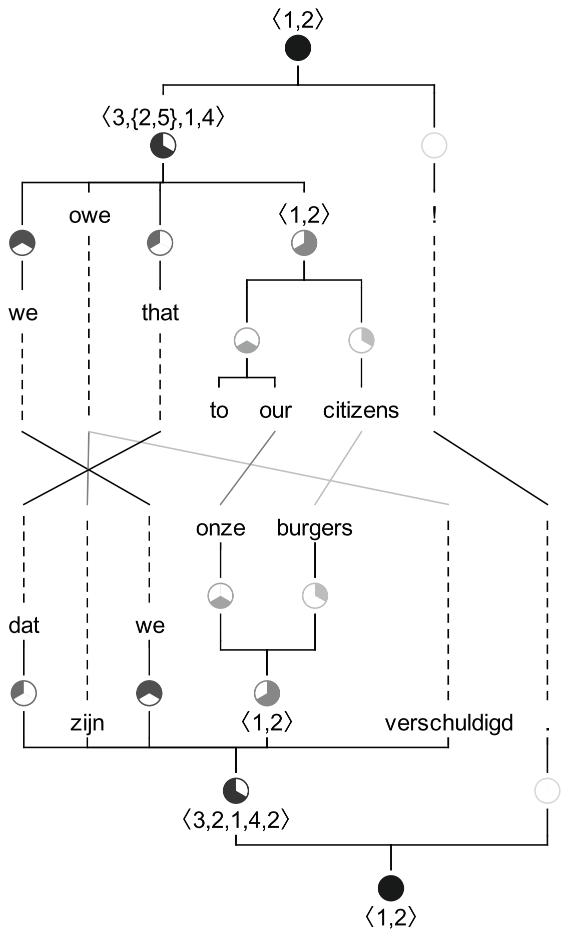


Figure 16: Dual HAT representations (for source and target sides) for a complex word alignment

of the English linked node, thereby maintaining the lexical word alignment for such cases within the HAT structure.

5.7 What are empirical word alignments?

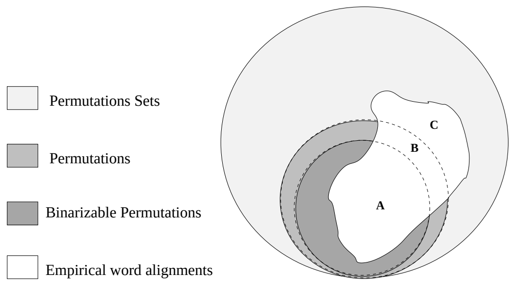


Figure 17: A sketch of a possible characterization of empirical word alignments

Figure 17 shows a sketch of a likely situation: empirical word alignments are not a proper subset of binarizable permutations, nor are they a proper subset of permutations. However, by definition, word alignments are a subset of s-permutations. The figure shows that empirical word alignments overlap with binarizable permutations (area A), with permutations (area A+B) and with s-permutations (area A+B+C). What are the relative sizes of areas A, B and C? Clearly this is an empirical question that depends on the nature of the parallel corpus (language pair and language use) and on the kind of word alignments found in it. In the next section we aim to explore this question empirically on some standard data sets with two kinds of well known word alignments.

6 Empirical explorations of recursive reordering in word alignments

Broadly speaking, the question addressed in the present section is

> What percentage of word alignments and translation equivalents can be represented by specifying formal constraints on the allowed subset of HATs (e.g., upperbound on branching factor, bucketting of node transduction operators as a "distance measure" from pure permutations)?

We explore this question on manually produced word alignments as well as automatically obtained word alignments. We note here that our empirical findings must be interpreted *in relation to the specific definition of sub-permutations, i.e., word aligned phrase pairs*.

Because HATs are minimally branching STPs for the word alignments, a primary differentiating parameter for the performance measures is the *maximal branching factor*:

β_{max} the *maximal branching factor* of the linked nodes (above the pre-terminal level) in the HATs. The branching factor is measured on both sides and all linked nodes in the HATs for a word alignment must fall within the range $[1..\beta_{max}]$ to be counted in.[9]

For a given β_{max} value we report:

Alignment coverage This is the percentage of word alignments for which the branching factor of all nodes in all HATs fall within the settings β_{max}. Alignment coverage is not necessarily equivalent to *alignment reachability* [Søgaard and Wu, 2009, Søgaard, 2010] or the complementary measure of *parsing failure rate (PFR)* [Zens and Ney, 2003, Søgaard and Wu, 2009], which are both reported under subsets of ITG. Alignment coverage is equivalent to alignment reachability under a given grammar formalism iff both are measured in under the same semantics and such that the setting of β_{max} correspond exactly to the formalism in question.[10] In this sense, alignment coverage for $\beta_{max} = 2$ is an exact (as opposed to upper/lower-bounds) on alignment reachability for NF-ITG.[11]

Translation equivalents coverage (TEC) This is the *percentage of phrase-pairs* that are represented by a linked node in the HATs (if any) that have maximal branching factor β_{max}. The TEC measure is strongly related to Translation Units Error Rate (TUER) [Søgaard and Kuhn, 2009, Søgaard and Wu, 2009]. In fact, if we use the same definition of translation equivalents, the same counting algorithm and the same representation for both (NF-ITG), we find that TUER = (1-TEC).

Binarizability score The ratio of the number of linked nodes in a HAT (subject to β_{max}) built for a given word alignment relative to the number of such nodes in a hypothetical, fully binary branching

[9]Null aligned words are not counted in the branching factor, i.e, even unaligned word dominated directly by a node contributes zero to the branching factor of that node. The rationale behind this is that we do not want to discriminate between s-permutations differing only by NULL aligned words. This means that the reported results are rather conservative.

[10]Unlike for ITGs, the case $\beta_{max} = 3$ is not necessarily equivalent to $\beta_{max} = 2$ because the s-permutations (as opposed to permutations) over three positions can be non-binarizable, see e.g., Figure 15.

[11]Because our algorithm builds HATs over minimal phrase pairs, the case $\beta_{max} = 2$ is equivalent to the NF-ITG over these minimal phrase pairs, i.e., given the defined semantics of word alignments.

HAT (using the length of the shortest among the source and target sentences minus one). The binarizability score, as opposed to TEC, provides a relatively objective measure of how hard it is to represent the word alignments in the corpus by deeply nested HATs. The lower the binarizability score, the less linked nodes exist in the HATs admitted (given β_{max}), if any at all. HATs that contain flat structures indicate complex word alignments consisting of unaligned words and many-to-many alignments that cannot be decomposed into smaller TE units. This can be seen to indicate idiomatic translation, as opposed to compositional translation (given the semantics defined here).

Because for some alignments our implementation can consume large memory in calculating the HAT forest we had to abort it in a small percentage of cases ($\leq 10^{-5}$), mostly very long alignments with many unaligned words. For all reported experiments we set a cut-off (set at 100k) on the number of inferences per linked node in the CYK chart implementing the HAT algorithm in Figure 12. Once the algorithm exceeds 100k inferences for a node we skip the alignment. In what follows the number of skipped sentence pairs is reported for each experiment separately.

6.1 Manual word alignments

In this part, we use the manually aligned part of the Hansards corpus (English-French), created and first used by Och and Ney [Och and Ney, 2000, Och and Ney, 2003]. This tiny corpus consists of 447 manually aligned sentence pairs, were alignment links are labeled with *Sure* or *Possible*. Some basic statistics of this corpus is shown in Table 1. We report the results for the Sure+Possible links, i.e., all alignment links.

Language Pair	**English-French**
Total number of sentence pairs	447
#skipped sentence pairs	0
#sentence pairs containing nulls	231
Mean source length	15.705±6.994
Mean target length	17.362±7.554
Mean and STD of ratio source to target lengths	0.928±0.221
Mean #links per word	2.52 ± 1.73

Table 1: Statistics of Hansards manually aligned corpus. The "mean #links per word" is calculated using the mean $\frac{\text{\# alignment links}}{\min_{x \in \{\mathbf{s},\mathbf{t}\}} length\ of\ x}$ over the corpus word alignments.

Tables 2a, 2b and 2c report respectively the break-down of coverage, TEC and binarizability score to β_{max}. All scores increase rapidly in the range $\beta_{max} \in [2..10]$, but more gradually for $\beta_{max} > 10$. The alignment coverage and the TEC results start low in the seventies for $\beta_{max} = 2$ (NF-ITG), but increase rapidly to the low nineties by $\beta_{max} = 6$. The increase continues at a decaying pace for higher values of β_{max}. The binarizability scores, Table 2c, are a different matter. The results make clear that the HATs built for the word alignments contain only at most 62% of the number of linked nodes in a *hypothetical* fully binary HAT (binarizable permutation). This suggests that these word alignments are represented with HATs that are somewhat flat relative to the hypothetical fully binary HATs. This observation completes the picture drawn by the coverage and TEC results for $\beta_{max} > 2$: the word alignments are clearly far more complex than the entry-level cases of binarizable permutations.

At least as interesting, Table 3 reports the coverage of word alignments to the kind of s-permutation (HATs) involved: binarizable permutations (BITTs), permutations (PETs) and s-permutations (HATs). Remarkably, merely moving away from binarizable to all permutations does not increase the coverage much, whereas the general case of s-permutations provides full coverage. This result together with the break-down of the statistics to β_{max} values in the other tables suggests actually that the cases of

	English-French
$\beta_{max} = 2$	71.46%
$\beta_{max} = 3$	77.08%
$\beta_{max} = 4$	82.47%
$\beta_{max} = 5$	87.19%
$\beta_{max} = 6$	90.11%
$\beta_{max} = 7$	92.58%
$\beta_{max} = 8$	94.38%
$\beta_{max} = 9$	96.40%
$\beta_{max} = 10$	97.75%
$\beta_{max} = 15$	99.33%
$\beta_{max} = 19$	100.00%

(a) Alignment coverage.

β_{max}	English-French
2	73.54%
3	80.66%
4	87.24%
5	91.74%
6	93.81%
7	95.12%
8	96.96%
9	98.30%
10	98.89%
15	99.81%
19	100.00%

(b) Translation Equivalents Coverage.

β_{max}	English-French
2	42.34%
3	46.97%
4	51.53%
5	55.00%
6	56.82%
7	58.19%
8	59.77%
9	61.02%
10	61.77%
15	62.66%
19	62.91%

(c) Binarizability scores.

Table 2: Scores for Sure+Possible manual alignments in the Hansards corpus as a function of β_{max}.

Kind of HATs (S-permutations)	English-French
BITTs (Binarizable permutations)	71.46%
PETs (All permutations)	72.14%
HATs (S-permutations)	100.00%

Table 3: The ratio of the different subsets of HATs in the manual Sure+Possible alignments in the Hansards corpus: BITTs, PETs and HATs

non-binarizable permutations tend to cooccur with other complex forms of alignments, including discontinuous and many-to-many cases. This does not mean that we do not need the full descriptive power of permutations, but that on their own they are *almost as insufficient as their binarizable subset* for capturing word alignments found in actual translation data.

	English-French
$D(\mathbf{a}, \mathbf{s}, \mathbf{t}) = 0$ (pure permutations)	72.04%
$D(\mathbf{a}, \mathbf{s}, \mathbf{t}) = 1$	77.85%
$D(\mathbf{a}, \mathbf{s}, \mathbf{t}) = 4$	85.46%
$D(\mathbf{a}, \mathbf{s}, \mathbf{t}) = 12$	90.16%
$D(\mathbf{a}, \mathbf{s}, \mathbf{t}) = 30$	95.53%
$D(\mathbf{a}, \mathbf{s}, \mathbf{t}) = 50$	98.66%
$D(\mathbf{a}, \mathbf{s}, \mathbf{t}) = 83$	100.00%

Table 4: Coverage of the Hansard manually aligned corpus as a function of $D(\mathbf{a}, \mathbf{s}, \mathbf{t})$.

To obtain a better picture, we refine Table 3 by ordering *s-permutations* into increasing subsets: *we measure how far a word alignment is from being a permutation*. Formally, a s-permutation (word alignment) differs from a permutation in that it may contain many more links than the bijective case and/or the source and target sides may contain a different number of positions. Consequently, we define for every word alignment $\langle \mathbf{s}, \mathbf{t}, \mathbf{a} \rangle$ its "distance" $D(\mathbf{a}, \mathbf{s}, \mathbf{t})$ from being a permutation simply as the absolute value of the difference between the number of individual alignment links it contains ($number\ of\ links(\mathbf{a})$) and the *minimum* number of positions ($length()$) between the source and target sides, where both numbers are measured over the minimal phrase pairs (not the individual words):

$$D(\mathbf{a}, \mathbf{s}, \mathbf{t}) \ = |number\ of\ links(\mathbf{a}) - \min_{x \in \mathbf{s},\mathbf{t}} length(x)|$$

This measure is a first approximation of the idea, but it provides a meaningful approximate break-down of the space between word alignments (s-permutations) and permutations. Table 4 shows the breakdown of the coverage of word alignments to any value of $D(\mathbf{a}, \mathbf{s}, \mathbf{t})$ in between the two extreme cases of pure permutations ($D(\mathbf{a}, \mathbf{s}, \mathbf{t}) = 0$) and arbitrary s-permutations that are equivalent to the word alignments themselves. Clearly, approximately one in five of these manual alignments falls somewhat far off from pure permutations ($D(\mathbf{a}, \mathbf{s}, \mathbf{t}) > 1$), whereas almost one in four is beyond binarizable permutations. This highlights the need for future refinements of s-permutations into more tight subsets that characterize manual word alignments.

6.2 Automatic word alignments

We report empirical results on English-Dutch, English-French and English-German corpora obtained from the respective corpora in the Europarl collection [Koehn, 2005] by setting an upperbound of 40 words on the sentence length on the source and target sides. Table 5 lists the sizes of the corpora. The corpora we use are about half the size of the original corpora but we think that they are large enough to contain word alignments Representative of the subsuming, full-size corpora.

Language Pairs	**English-Dutch**	**English-French**	**English-German**
Total number of sentence pairs	945167	949408	995909
#skipped sentence pairs	745	257	453
#sentence pairs containing nulls	801948	783367	839335
Mean source length	21.295±8.916	20.556±8.585	21.559±9.138
Mean target length	21.212±9.011	22.552±9.383	20.459±8.872
Mean and STD of ratio of source to target lengths	1.029±0.219	0.934±0.201	1.081±0.244
Mean & STD #links per word	1.12±0.14	1.15±0.15	1.14±0.16

Table 5: The corpora used in our analysis (sentence length ≤ 40 words). The "mean #links per word" is calculated using as the mean over the corpus alignments for the ratio $\frac{\text{\# alignment links}}{\min_{x \in \{\mathbf{s},\mathbf{t}\}} length\ of\ x}$.

Following standard practice (e.g., [Koehn et al., 2007]), the sentences were lower-cased and tokenized using the relevant Moses scripts.[12] The sentence pairs were word aligned using GIZA++[13] with number of training iterations as follows: four for IBM model 1, three for IBM model 3, four for IBM model 4 and three for HMM model. The grow-diag-final heuristic was used for symmetrization of the alignments in the two translation directions.

Tables 6 and 7 show the alignment and TE Coverage results respectively. Unlike the manual word alignments, for the automatic alignments the coverage and TEC results increase dramatically for β_{max} values in $[2..6]$. For $\beta_{max} = 2$ the results are in the mid/low forties for English-Dutch/German and low fifties for English-French, but by $\beta_{max} = 6$ all results are in the nineties or thereabout.

Automatic alignments obtained by symmetrization heuristics (particularly grow-diag-final) are built in a way that allows the extraction of a large number of phrase pair equivalents. This could explain why binary branching HATs (BITTs) have lower coverage of such word alignments, and why more often than not a larger branching factor than two is needed. Table 8 supports this observation. On the one hand, for at most binary branching HATs ($\beta_{max} = 2$), the binarizability score is very low in the

[12]`http://www.statmt.org/moses/`

[13]`http://www-i6.informatik.rwth-aachen.de/Colleagues/och/software/GIZA++.html.`

	English-Dutch	English-French	English-German
$\beta_{max} = 2$	45.52%	52.84%	45.60%
$\beta_{max} = 3$	55.77%	67.27%	56.80%
$\beta_{max} = 4$	73.95%	82.60%	74.75%
$\beta_{max} = 5$	83.51%	90.01%	84.63%
$\beta_{max} = 6$	89.92%	94.40%	90.76%
$\beta_{max} = 7$	93.56%	96.72%	94.36%
$\beta_{max} = 8$	95.91%	98.06%	96.55%
$\beta_{max} = 9$	97.37%	98.85%	97.88%
$\beta_{max} = 10$	98.33%	99.31%	98.71%
$\beta_{max} = 15$	99.86%	99.95%	99.90%
$\beta_{max} = 21$	100.00%	100.00%	100.00%

Table 6: Coverage of the corpus as a function of β_{max} for symmetrized (grow-diag-final) word alignments in EuroParl parallel corpora (sentence length ≤ 40) for three language pairs

	English-Dutch	English-French	English-German
$\beta_{max} = 2$	44.63%	51.99%	43.40%
$\beta_{max} = 3$	56.35%	68.55%	56.13%
$\beta_{max} = 4$	75.79%	84.64%	75.46%
$\beta_{max} = 5$	85.51%	91.93%	85.79%
$\beta_{max} = 6$	91.64%	95.83%	91.83%
$\beta_{max} = 7$	94.91%	97.71%	95.23%
$\beta_{max} = 8$	96.90%	98.73%	97.18%
$\beta_{max} = 9$	98.08%	99.28%	98.33%
$\beta_{max} = 10$	98.82%	99.59%	99.02%
$\beta_{max} = 15$	99.91%	99.97%	99.94%
$\beta_{max} = 20$	100.00%	100.00%	100.00%

Table 7: Translation Equivalents Coverage as a function of β_{max}

	English-Dutch	English-French	English-German
$\beta_{max} = 2$	33.45%	41.43%	32.75%
$\beta_{max} = 3$	42.40%	54.75%	42.45%
$\beta_{max} = 4$	58.38%	68.92%	58.19%
$\beta_{max} = 5$	66.47%	75.47%	66.59%
$\beta_{max} = 6$	71.75%	79.21%	71.62%
$\beta_{max} = 7$	74.63%	81.10%	74.48%
$\beta_{max} = 8$	76.43%	82.15%	76.14%
$\beta_{max} = 9$	77.52%	82.76%	77.14%
$\beta_{max} = 10$	78.22%	83.10%	77.74%
$\beta_{max} = 15$	79.27%	83.54%	78.57%
$\beta_{max} = 21$	79.35%	83.57%	78.62%

Table 8: Binarizability scores as a function of β_{max}

Kind of HATs (S-permutations)	English-Dutch	English-French	English-German
$BITTs$ (Binarizable permutations)	45.52%	52.84%	45.60%
$PETs$ (Permutations)	52.63%	56.56%	52.55%
$HATs$ (S-permutations)	100.00%	100.00%	100.00%

Table 9: The ratio of the different subsets of HATs in the corpus: BITTs, PETs and HATs

thirties (Dutch/German) or forties (French) suggesting that these word alignments are hard to capture with BITTs. On the other, by $\beta_{max} \geq 10$ the binarizability score is around the 83% for English-French suggesting HATs with many linked nodes, particularly when we contrast this with 62% for the Hansards manual alignment. It is unlikely that the latter difference can fully be explained by the difference in language use (Hansards vs EuroParl) and, in fact, the shorter average sentence length in the Hansards manually aligned corpus actually suggests the reverse situation should be true. This supports the hypothesis that the symmetrized automatic alignments are built such that they can facilitate extracting a larger number of phrase pair equivalents, leading to many more nodes in the HATs than manual alignments.

Table 9 shows that the coverage of BITTs around 52% for French and 45% for Dutch and German. The coverage of PETs (permutations) increases by 4-7% only, again suggesting that neither BITTs nor PETs (as pure permutation-devices) can provide good coverage of phenomena in word alignments. If only approximately 50% of all such word alignments can be represented fully as a permutation, then the other 50% demands the notion of a s-permutation that can capture discontinuous alignments and complex many-to-many cases. Yet, s-permutations remain simple extensions of permutations and HATs can be seen as conservative extensions of PETs and BITTs.

	English-Dutch	English-French	English-German
$D(\mathbf{a}, \mathbf{s}, \mathbf{t}) = 0$ (pure permutations)	52.63%	56.55%	52.55%
$D(\mathbf{a}, \mathbf{s}, \mathbf{t}) = 1$	75.86%	80.12%	75.29%
$D(\mathbf{a}, \mathbf{s}, \mathbf{t}) = 2$	90.11%	92.27%	89.28%
$D(\mathbf{a}, \mathbf{s}, \mathbf{t}) = 3$	96.05%	97.18%	95.60%
$D(\mathbf{a}, \mathbf{s}, \mathbf{t}) = 4$	98.44%	98.95%	98.19%
$D(\mathbf{a}, \mathbf{s}, \mathbf{t}) = 5$	99.39%	99.60%	99.28%
$D(\mathbf{a}, \mathbf{s}, \mathbf{t}) = 10$	100.00%	100.00%	100.00%

Table 10: Coverage of the corpus as a function of $D(\mathbf{a}, \mathbf{s}, \mathbf{t})$.

Table 10 shows the breakdown of the coverage statistic of automatic word alignments to $D(\mathbf{a}, \mathbf{s}, \mathbf{t})$, i.e., the measure of "distance" of the word alignment from being a pure permutation.

In symmetrized automatic alignments, it turns out that almost one half is beyond permutations, whereas by distance $D(\mathbf{a}, \mathbf{s}, \mathbf{t}) \leq 5$ there is almost full coverage. Intuitively speaking, only a small number of deletions of alignment links in these alignments should result in alignments that resemble permutations. This does not reveal the full complexity of these alignments but it does suggest that the automatic alignment that are non-binarizable are less complex than their Hansards manual (Sure+Possible) counterpart. We think that this is because the Hansards manual alignments (Sure+Possible) are in many cases dense with links (see the "mean number of links per word" in Tables 1 and 5). Some of these manual alignments simply link almost all source words with almost all target words leading to rather flat HATs and fewer phrase pair translation equivalents than the automatic alignments.

6.3 Discussion and related empirical work on the analysis of alignments

The empirical results presented in the preceding section relate to a debate concerning the representation power of (Normal-Form) ITG for translation data. The existing reports concentrate mostly on *upper bounds* for the representation power of ITG in terms of manual or automatic word alignments [Zens and Ney, 2003, Galley et al., 2004, Wellington et al., 2006, Søgaard and Wu, 2009, Søgaard, 2010]. A small part of our empirical results can be seen to contribute to this debate, particularly the case concerning BITTs. Søgaard and Wu [Søgaard and Wu, 2009] observe that the reports in the literature differ considerably along various dimensions making it difficult to compare coverage results. Indeed, in studying the existing literature we found it particularly difficult to pin down the exact choices made, which makes it difficult to interpret the reported results. There is, however, a good reason for the difficulty of exact description as we explain next.

We think that the problem of measuring the exact coverage of word alignments by a synchronous grammar is particularly complicated because there is no a priori, objective, formal grounds on which word alignments and synchronous grammars can be formally intersected (or composed). Before intersecting these two completely different representations, it is necessary to specify a *shared meaning/semantics*. In this work we define for every word alignment a semantically equivalent set of HATs, and then we check that these HATs can be built by an instance of the grammar formalism, i.e., that there exists an instance at all that can generate these HATs. As defined earlier, for measuring sentence-level coverage, we check that there exists an instance of the grammar formalism that can generate all these HATs exactly. And for coverage/recall of translation equivalents/units (TEC), we check the percentage of linked nodes that can be generated by an instance of the grammar formalism.

Our results with $\beta_{max} = 2$ are in fact coverage and TEC results for NF-ITG. Some earlier work has concentrated on measuring *upperbounds* on the coverage/TEC either by discounting the complex alignment cases that cannot be covered by NF-ITG, e.g., [Søgaard and Kuhn, 2009, Søgaard and Wu, 2009], or by defining a "lighter" semantics of word alignments by employing a disjunctive interpretation (as opposed to the more accepted conjunctive interpretation) [Wellington et al., 2006].

As far as we can see, the results presented in [Zens and Ney, 2003, Wu et al., 2006, Søgaard, 2010], although based on a different approach and pertain to different data sets and word alignments, are measured in ways that are close to implementing the intersection used here. Zens and Ney [Zens and Ney, 2003] employ Viterbi alignments on Hansards data (sentence length up to 30 words) and obtain far higher coverage results for NF-ITG ($\approx$ 81% and 73% depending on direction) than our results for English-French EuroParl data with symmetrized word alignments ($\approx$ 53%). Apart from differences in corpus data, symmetrized alignments, constituting the backbone data for training state-of-the-art systems, are known to be distinctively different from their Viterbi uni-directional ancestors. The coverage result of [Søgaard, 2010] on the manual Hansards data (77%) comes very close to our coverage result (72%). Søgaard [Søgaard, 2010] is presented densely and somewhat informally that some details escape us. We attribute the difference to various reasons, including sentence-length differences (in [Søgaard, 2010] the cutoff is 15 words) and choices of how to define translation equivalents with unaligned words on either side. The work of [Wu et al., 2006] concerns Arabic-English data, which is not studied here.

A completely distinct work that reports measures of word alignment coverage under ITG constraints is [Huang et al., 2009]. The work is based on the GHKM [Galley et al., 2004] method of extracting synchronous rules, which involves target-language syntax. The authors report percentages of binarizable synchronous rules extracted from the word alignments. The results reported are incomparable to our results for NF-ITG because they are subject to the GHKM extraction method of synchronous rules, which encapsulate very difficult word alignments as internal, lexical parts of a synchronous rule. By doing so, the coverage is measured with regards to a different semantics (the GHKM extraction method) of word alignments than our choice of semantics. Our semantics of word alignments is more exhaustive than the GHKM semantics in that we allow all phrase pairs to be extracted without constraints from monolingual syntax or other performance-driven constraints.

7 Conclusions and future work

We contributed an algorithm to factorize general word alignments into unique sets of synchronous trees represented as packed forests in the same fashion known from monolingual parsing. Viewing word alignments from the perspective of word order differences, initially we extended permutations into s-permutations that constitute asymmetric representations of word alignments. Subsequently we identified sub-permutations of s-permutations as word aligned phrase pairs, and developed our factorization algorithm to represent the sub-permutations in Hierarchical Alignment Trees (HATs). The HAT representation, a conservative extension of the synchronous trees known from ITG, posses some attractive properties, particularly that every node in every HAT is minimally branching and decorated with a transduction operator.

On the empirical side we used our HAT algorithm to analyze manual and automatically acquired word alignments. Our analysis concentrates on a break-down of statistics of word alignments by the maximal branching factor needed in the HATs that represent them. The complexity of the transduction operators can be useful for dividing word alignments into sub-classes as exemplified in the empirical part of this work. As a limited sub-study, we also report exact coverage results of word alignments for normal-form ITG.

In future work we plan to explore different semantics for word alignments (beyond phrase pairs) and different definitions of STPs beyond bijective layered linking. Besides studying the exact coverage of ITGs and other synchronous grammars, we expect that our HAT representation can be used to shed light on the stability of the Direct Correspondence Assumption of monolingual syntactic representations projected using word alignments, e.g., [Fox, 2002, Hwa et al., 2002]. Finally, we hope that this study prepares the ground for novel and useful methods for the automatic learning of hierarchical alignments in parallel corpora, the original topic that started off this study.

8 Acknowledgments

We thank three anonymous reviewers for pointing us to crucial references to strongly related work and important suggestings for improvement. This research was supported by VIDI grant 639.022.604 and Free Competition Exact Sciences grant 612.066.929, both from The Netherlands Organization for Scientific Research (NWO).

References

[Blunsom et al., 2008] Blunsom, P., Cohn, T., and Osborne, M. (2008). Bayesian synchronous grammar induction. In *NIPS*, pages 161–168.

[Brown et al., 1990] Brown, P., Cocke, J., Pietra, S. D., Pietra, V. D., Jelinek, F., Lafferty, J. D., Mercer, R., and Roossin, P. S. (1990). A statistical approach to machine translation. *Computational Linguistics*, 16(2):79–85.

[Bui-Xuan et al., 2005] Bui-Xuan, B.-M., Habib, M., and Paul, C. (2005). Revisiting t. uno and m. yagiura's algorithm. In *Proceedings Algorithms and Computation, 16th International Symposium, ISAAC 2005, Sanya*, pages 146–155.

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

[Dijkstra, 1959] Dijkstra, E. W. (1959). A note on two problems in connexion with graphs. *Numerische Mathematik*, 1:269–271.

[Eisner, 2003] Eisner, J. (2003). Learning non-isomorphic tree mappings for machine translation. In *Proceedings of ACL 2003 (companion), 41st Annual Meeting of the Association for Computational Linguistics, Companion Volume to the Proceedings, 7-12 July 2003, Sapporo Convention Center, Sapporo, Japan*, pages 205–208. The Association for Computational Linguistics.

[Fox, 2002] Fox, H. J. (2002). Phrasal cohesion and statistical machine translation. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing - Volume 10*, EMNLP '02, pages 304–311, Stroudsburg, PA, USA. Association for Computational Linguistics.

[Galley et al., 2004] Galley, M., Hopkins, M., Knight, K., and Marcu, D. (2004). What's in a translation rule? In *Proceedings of the HLT/NAACL-04*.

[Gildea et al., 2006] Gildea, D., Satta, G., and Zhang, H. (2006). Factoring synchronous grammars by sorting. In *ACL*.

[Haghighi et al., 2009] Haghighi, A., Blitzer, J., DeNero, J., and Klein, D. (2009). Better word alignments with supervised itg models. 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 923–931, Suntec, Singapore. Association for Computational Linguistics.

[Huang et al., 2009] Huang, L., Zhang, H., Gildea, D., and Knight, K. (2009). Binarization of synchronous context-free grammars. *Computational Linguistics*, 35(4):559–595.

[Hwa et al., 2002] Hwa, R., Resnik, P., Weinberg, A., and Kolak, O. (2002). Evaluating translational correspondence using annotation projection. In *Proceedings of the Annual Meeting of Association of Computational Linguistics (ACL 2002)*, pages 392–399.

[Janssen, 1998] Janssen, T. M. V. (1998). An overview of compositional translations. In *Revised Lectures from the International Symposium on Compositionality: The Significant Difference*, COMPOS'97, pages 327–349, London, UK. Springer-Verlag.

[Koehn, 2005] Koehn, P. (2005). Europarl: A parallel corpus for statistical machine translation. In *Proceedings of the MT Summit X*, pages 79–86.

[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 ACL'07*, pages 177–180, Prague, Czech Republic.

[Koehn et al., 2003] Koehn, P., Och, F., and Marcu, D. (2003). Statistical phrase-based machine translation. In *In proceedings of Human Language Technologies and the North-American Chapter of the ACL (HLT-NAACL)*, pages 48–54, Edmonton, Canada.

[Landau et al., 2005] Landau, G. M., Parida, L., and Weimann, O. (2005). Gene proximity analysis across whole genomes via pq trees^1. *Journal of Computational Biology*, 12(10):1289–1306.

[Landsbergen, 1982] Landsbergen, J. (1982). Machine translation based on logically isomorphic montague grammars. In *Proceedings of the 9th conference on Computational linguistics - Volume 1*, COLING '82, pages 175–181, Czechoslovakia. Academia Praha.

[Mylonakis and Sima'an, 2010] Mylonakis, M. and Sima'an, K. (2010). Learning probabilistic synchronous cfgs for phrase-based translation. In *Fourteenth Conference on Computational Natural Language Learning*, Uppsala, Sweden.

[Och and Ney, 2000] Och, F. and Ney, H. (2000). Improved statistical alignment models. In *Proceedings of the the 38th Annual Meeting on Association for Computational Linguistics (ACL)*, pages 440–447.

[Och and Ney, 2003] Och, F. and Ney, H. (2003). A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29(1):19–51.

[Poutsma, 2000] Poutsma, A. (2000). Data-oriented translation. In *Proceedings of the International Conference on Computational Linguistics (COLING'00)*, pages 635–641.

[Sima'an, 1999] Sima'an, K. (1999). *Learning Efficient Disambiguation*. PhD dissertation (University of Utrecht). ILLC dissertation series 1999-02, University of Amsterdam, Amsterdam.

[Søgaard, 2010] Søgaard, A. (2010). Can inversion transduction grammars generate hand alignments? In *Proccedings of the 14th Annual Conference of the European Association for Machine Translation (EAMT)*.

[Søgaard and Kuhn, 2009] Søgaard, A. and Kuhn, J. (2009). Empirical lower bounds on alignment error rates in syntax-based machine translation. In *Proceedings of the Third Workshop on Syntax and Structure in Statistical Translation*, SSST '09, pages 19–27, Stroudsburg, PA, USA. Association for Computational Linguistics.

[Søgaard and Wu, 2009] Søgaard, A. and Wu, D. (2009). Empirical lower bounds on translation unit error rate for the full class of inversion transduction grammars. In *Proceedings of the 11th International Workshop on Parsing Technologies (IWPT-2009), 7-9 October 2009, Paris, France*, pages 33–36. The Association for Computational Linguistics.

[Tinsley et al., 2009] Tinsley, J., Hearne, M., and Way, A. (2009). Exploiting parallel treebanks to improve phrase-based statistical machine translation. In Gelbukh, A. F., editor, *Proceedings of the Computational Linguistics and Intelligent Text Processing, 10th International Conference, (CICLing 2009)*, volume 5449 of *Lecture Notes in Computer Science*, pages 318–331. Springer.

[Uno and Yagiura, 2000] Uno, T. and Yagiura, M. (2000). Fast algorithms to enumerate all common intervals of two permutations. *Algorithmica*, 26(2):290–309.

[van Noord, 1995] van Noord, G. (1995). The intersection of finite state automata and definite clause grammars. In *Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics (ACL'95)*, pages 159–165.

[Wang et al., 2010] Wang, W., May, J., Knight, K., and Marcu, D. (2010). Re-structuring, re-labeling, and re-aligning for syntax-based machine translation. *Computational Linguistics*, 36:247–277.

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

[Wu, 1996] Wu, D. (1996). A polynomial-time algorithm for statistical machine translation. In *Proceedings of Annual Meeting of the Association for Computational Linguistics*, Santa Cruz.

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

[Wu, 2010] Wu, D. (2010). *Alignment. In CRC Handbook of Natural Language Processing, Second Edition*, pages 367–408. CRC Press.

[Wu et al., 2006] Wu, D., Carpuat, M., and Shen, Y. (2006). Inversion transduction grammar coverage of arabic-english word alignment for tree-structured statistical machine translation. In *Proceedings of the IEEE/ACL 2006 Workshop on Spoken Language Technology (SLT 2006)*.

[Wu and Wong, 1998] Wu, D. and Wong, H. (1998). Machine translation wih a stochastic grammatical channel. In *Proceedings of ACL-COLING'98*, pages 1408–1415, Columbus, OH, USA.

[Younger, 1967] Younger, D. (1967). Recognition and parsing of context-free languages in time n^3. *Information and Control*, 10(2):189–208.

[Zens and Ney, 2003] Zens, R. and Ney, H. (2003). A comparative study on reordering constraints in statistical machine translation. In *Proceedings of the Annual Meeting of the ACL*, pages 144–151.

[Zens et al., 2002] Zens, R., Och, F., and Ney, H. (2002). Phrase-based statistical machine translation. In *Proceedings of KI: Advances in Artificial Intelligence*, pages 18–32.

[Zhang and Gildea, 2007a] Zhang, H. and Gildea, D. (2007a). Enumeration of factorizable multi-dimensional permutations. *Journal of Integer Sequences*, 07(5.8).

[Zhang and Gildea, 2007b] Zhang, H. and Gildea, D. (2007b). 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 *COLING*, pages 1081–1088.

[Zhang et al., 2006] Zhang, H., Huang, L., Gildea, D., and Knight, K. (2006). Synchronous binarization for machine translation. In *HLT-NAACL*.

[Zhechev and Way, 2008] Zhechev, V. and Way, A. (2008). Automatic generation of parallel treebanks. In Scott, D. and Uszkoreit, H., editors, *Proceedings of the 22nd International Conference on Computational Linguistics (COLING 2008)*, pages 1105–1112.