The Induction and Evaluation of Word Order Rules using Corpora based on the Two Concepts of Topological Models

Bernd Bohnet
Innovative Language Technology, TTI GmbH
and
University of Stuttgart
Visualization and Interactive Systems Group
Universitätstr. 58, 70569 Stuttgart, Germany
bohnet@informatik.uni-stuttgart.de

Abstract

Using dependency trees in natural language generation and machine translation raise the need to derive the word order from dependency trees. This task is difficult for languages with (partly) free word order and comparatively easier for languages with fixed word order. This paper describe (**a**) the two basic elements of topological models, (**b**) rule patterns for the mapping of dependency trees to topological trees, (**c**) the automatic acquisition of word order rules from corpora, annotated with dependency structures and (**d**) an approach for the automatic evaluation of the results.

1 Introduction

Dependency structures are frequently used in Natural Language Generation (NLG) and in some cases in Machine Translation (MT). In NLG, dependency structures are used in the surface realization step. The input to the surface realizer is defined by the standard architecture RAGS for NLG systems as "syntactic representations" which is "based on some notion of abstract syntactic structure which does not encode surface constituency or word order", cf. page 17 (Mellish et al., 2006). These input structures are usually dependency structures.

In MT, statistical n-gram approaches are rather successful and therefore mostly used. Nevertheless, some systems use dependency trees in order to improve the results for instance Lavoie et al. (2000), Čmejrek et al. (2003), Ding and Palmer (2005) and some approaches describe more the theoretical fundament for improving the translation results like Mel'čuk and Wanner (2006).

We will give a brief overview over the history of topological models, since topological models are the basis for this paper. The first model that was developed was the German topological model, cf. Drach (1937), Bech (1955), and Höhle (1986). The complexity of the German word order is obviously the reason behind the development of the German topological model. The syntax and the information structure determines the positions of the constituents in the model. The most salient parts are the so called sentence brackets, namely the *left bracket* (LK) and the *right bracket* (RK). The left bracket usually contains the finite verb or a conjunction. The right bracket can be empty or it can contain the infinite verb(s).

John	[wird]$_{LK}$	nach Berlin	[reisen]$_{RK}$.	
'John	will	to Berlin.	travel'$_{lit.}$	
[weil]$_{LK}$	es	ihm	[gefällt]$_{RK}$.	
'because	it	he	likes. '$_{lit.}$	

The syntax determines the constituents of the brackets. In relation to the brackets, three positions are possible: before the left bracket (pre-field), in the middle, i.e., between the left and the right brackets (middle field), and after the right bracket (post-field).

[Nach Berlin]$_{VF}$	[ist]$_{LK}$	[[er] [um 5]]$_{MF}$	[abgereist]$_{RK}$.
'to Berlin	is	John at 5am	departed'$_{lit.}$

This kind of approach has been developed only with regard to German and a few other languages, such as ancient French by Skarup (1975) and

Warlpiri by Donohue and Sag (1999). For some languages, such as English, the word order is easy to derive and therefore, only a simple topological model is needed. For instance, phrase structures fulfil already the requirements for a topological model with regard to English as well as with regard to quite a few other languages. Phrase structures describe the syntax by inclusion of constituent in other constituents and the position within the constituents. In the following example on the left, *the* and *kitchen* are constituents within the constituent *c*. This can occur recursively, as shown in the example below.

$[\text{the}_a\ \text{kitchen}_b]_c$ $[\text{in}_a\ [\text{the kitchen}]_b]_c$

Figure 1 shows a dependency tree. The words of the dependency tree are not ordered. Therefore, the word order has to be derived. Xia and Palmer (2001) describe a method for converting dependency structures to phrase structures. The conversion grammar is trained on corpora annotated with dependency structures and phrase structures. In order to learn word order rules, this method can be simplified, so that only dependency structures and the word order is needed.

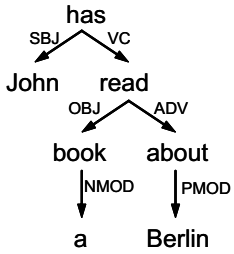

Figure 1: Dependency Tree.

Bohnet and Seniv (2004) introduced a method to map dependency structures to unordered phrase structures for languages with free word order. Since the constituents are not ordered, additional rules are needed to derive the order. For German, phrase structures do not allow to determine all possible orders, since for instance the phrases can be discontinuous. The classical German topological model describes the word order appropriately. Ideally, the dependency trees would be mapped directly to the topological model. Unfortunately, no large corpora annotated with topological fields are available. Therefore, we were forced to use phrase structures in order to learn German word order rules.

Fortunately, in the last years, corpora annotated with dependency structures for several languages have become available. This was encouraged by two "shared tasks" for dependency parsing. Thus, in the year 2006, 13 corpora became available, and in the year 2007 10 corpora partly identical became available. Out of these, we used to learn linearization grammars the Catalan corpus, the English Penn Treebank, the German Tiger corpus, and the Italian corpus .

2 The two Elements of Topological Models

The two elements of topological models can be discovered by analyzing various topological models, that have been developed mainly for text generation.

Bröker (1997) describes word order by domains and explicit precedence relations. The domains do not allow the words from outside of a domain to go between words contained in a domain. A topology is derived with rules based on modal logic from a dependency tree.

Duchier and Debusmann (2001) use a tree to describe the linear precedence (LP). The tree is projective and partially ordered. The edges are labelled with the names of topological fields. The LP tree is derived from a dependency tree called immediate dominance tree (ID). The LP tree is computed from the ID tree by a constrained based approach using lexical constraints and conditions for the claiming of nodes.

The topological model of Gerdes (2002) consist of topological fields and boxes. Boxes contain fields which contain recursively boxes and boxes again contain fields. The boxes and fields are supplied in form of a list. Both behave like domains. A topology is derived by traversing the dependency tree top down. Each time a word is placed in a box or field, depending on its subcatframe, new boxes or fields are created within the box or field.

Word order domains have been used also for the linearization of phrase structures, cf. (Reape, 1993), (Rambow, 1994), (Kathol and Pollard, 1995). Reape describes word order in terms of the containers which are associated with phrases. A container can include recursively other containers and words. For the mapping from phrase structures to such a container structure, the continuous phrases are associ-

ated with a container and the word of discontinuous phrases are included in the parent container. The order between the words is kept during the mapping.

The topological models above define word order either in terms of lists or sets. The sets need additional precedence relations for ordering the elements. A model using sets and precedence relations would allow to define models which use lists. The models using sets cannot be formulated directly in terms of lists, since models based on sets have the possibility of underspecification. Underspecified models contain partially ordered sets and represent several possible orders.

Domains and precedence relations are the most general concepts. Each of the previous sketched models can be decomposed using these concepts. In the following, we define formally, the two concepts of topological models.

Def. 1 *[Precedence Relation]*
An precedence relation defines an order between words or sets. Formally, an order relation is defined as a subset of the Cartesian product of a set W: $<\subseteq W \times W$*, where* $(w_i, w_j) \in <$ *if* w_i *is before* w_j

Def. 2 *[Domain[1]]*
A domain is a set of words and domains, where the elements of a domain, that contains another domain, are not present between the words of this other domain or recursively contained domains:
if $\exists D_m \in P_t : \forall\, x \in P_t \ \wedge\ x \neq D_m \rightarrow$
$(\forall y \in^* D_m : x < y) \vee (\forall y \in^* D_m : x > y)$. [2]

3 Rule Patterns and Mapping Procedure

For the definition of rule patterns and examples, graph transducers are used. Knight and Graehl (2005) give a good overview of different (tree) transducer types. We use transducer with two tapes, cf. (Bohnet, 2006). These transducers have many advantages over single tape transducer: Context is possible on both tapes, they are bidirectional, no ordering of rules is needed and a static relationship between symbols of the tapes is introduced. The most important feature is that the rules can be applied in parallel to all parts of the tree in contrast to normal tree transducers which traverse a tree top-down or bottom-up. Nevertheless, it is still possible to have an execution order between rules, then the order is determined by the rule context and not by manual ordering. The main advantage of applying rules in parallel is that this allows to define a grammar in correspondence to the independence of syntactic constituents as they exist in natural language. For instance, the complements and adjuncts of a verb can be ordered in parallel and independently from the adjuncts of a noun such as modifiers and determiners.

The topological models of the sentences are represented as hierarchical graphs. We use the hierarchical graph definition of Busatto (2002). He defines a hierarchical graph as an underlying flat graph and on top of this a directed acyclic graph *(DAG)* which represent the hierarchy.

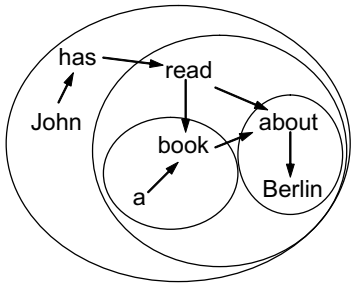

Figure 2: Precedence Graph.

The order relations and domains can be mapped directly to hierarchical graphs. The underlying flat graph represents the order relation and it is named G_O. The nodes of the graph G_O are the words which have to be ordered and the order relation as defined in Def. 1 is equal to the edge definition: $E_w \subseteq W \times W$. The domain and the recursive inclusion of domains builds the hierarchy and is represented by DAG G_D with the nodes D and the edges $E_D \subseteq D \times D$. The two graphs are coupled by a bipartite graph. A bipartite graph consists of two different node types. In this case, it links nodes from the words W and the domains D. The edges are defined as $E_c \subseteq D \times W$. The nodes, the edges and the domains can be labelled. Since we do not need

[1] We have chosen the name domain, since in the mathematic domain theory (Gierz et al., 2003) and order theory, the partly ordered sets are called domains, we use therefore, the same name for this basic element of topological word order models. Additionally, this name is already frequently used for this concept, cf. (Reape, 1993), (Kathol and Pollard, 1995), (Bröker, 1997).

[2] The definition means that if a domain D_m is in a domain P_t (its parent domain) then the other words or domains of the domain P_t are either before the words (and recursivley contained words) or after the words of the domain D_m.

the formal definition for the labels, we leave it out. Finally, the hierarchical graph is formed by the three graphs as defined above H = $\langle G_O, G_D, G_C \rangle$. An example of hierarchical graph that describes the word order of a sentence is shown in Figure 2. We call this type of a graph *precedence graph*.

Deriving the word order from a precedence graph. The word order of a precedence graph is derived in two steps. (1) If the graph has cycles then an edge from each of the cycles is removed. (2) A topological sort is applied to the precedence graph in order to derive the word order. A topological sort is a permutation p of the nodes of a directed graph such that an edge i, j implies that i appears before j in p. The topological sort is applied recursively to the domains. The domains are ordered by edges crossing domain borders.

Creating the precedence graph. The precedence graph is build by rules. Parts of the dependency tree build the left-hand side of rules and parts of the precedence graph build the right-hand side. The right-hand side of the rules are created without replacing the parts in the source graph. The created parts build a new graph as result. In this sense, the rules read from one tape and write to another one. After the creation of the parts defined on the right-hand sides, the resulting graph parts are not yet connected. In order to connect these parts, the correspondence links, like in the case of FST[3], can be introduced. Using this technique, they can either (1) grasp a part of the target graph and attach to this part a new part or (2) they can indicate which parts should be glued together. The first approach is useful when the rules are applied in a sequence. In this way already created parts can be accessed.

The advantage of this is that with (1) rules can be executed in a sequence and with (2) the rules can be applied independently in paralle. In this way, an execution sequence of rules is not forced by the processing technique and on the other hand, it is still possible as in the case of FST and in the case of tree transducer to have a sequence, but then justified by linguistic demands.

A linearization grammar has two tasks: building the domain hierarchy and ordering the nodes contained in the domains. Therefore, the rules fall into two types: domain rules and precedence rules. Domain rules consist of domain creation rules and domain adjunct rules. In the next paragraphs, we describe the rule patterns and a sample grammar.

Domain creation rules are used to define elementary domains. Figure 3 shows on the left a rule pattern which represents this rule type. The domain creation rules have on the left-hand side two nodes connected by an edge and on the right-hand side a domain containing two nodes. The nodes of the left-hand side and right-hand side are connected by correspondence links which are used to unify target nodes having a link to the same node in the source graph. These rules can build also domains containing more than two nodes. Overlapping domains are unified by the rule interpreter, if the labels are equal. The rule on the right shows an example. The example rule can be applied to the dependency tree shown in Figure 1. The left-hand side matches to the nodes labelled with *book* and *a* and creates a domain with these two nodes as shown in Figure 2. In order to build the topological model for this sentence, two additional rules of the same type are needed with the edge labelled SBJ and PMOD.[4] The rules might have additional conditions which take for instance into account the part of speech tag.

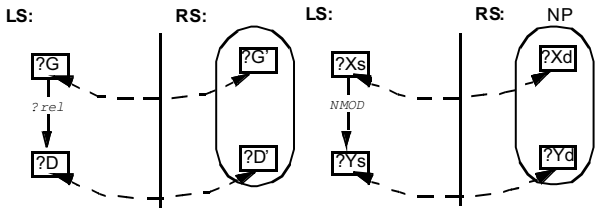

Figure 3: Domain creation rule pattern and example

Domain adjunct rules place domains into other domains. This rule type matches to an edge and grasps in the target graph a domain and places it in a

[3]FST rule consists of three parts; the correspondence, the rule operator and the environment or context, e.g. e:t => C:@ . The correspondence specifies a lexical symbol that corresponds to a surface symbol. The corresponds is indicated by a colon. We have taken up this idea and connect, in case of tree transducer, the left-hand side with the right-hand side by dashed lines instead of the colon. The operator of FST specifies the relationship between the correspondence and the context. The context specifies the environment in which the correspondence is found. While the lexical and surface symbols of FST are on the same side, the symbols of tree and graph transducers are usually separated on the left-hand side and right-hand side.

[4]A complete grammar also has rules with the edge labels OBJ and ADV. For the example these rules are not necessary.

newly created domain. This new domain might also be unified with domains created by domain creation rules. Figure 4 shows a rule pattern of this rule type. The dashed lines on the right-hand side indicate context in the target graph. The rules are applied in the second step because of the context. For the creation of English topological models, only two steps are needed. In the first step, all domain creation rules and precedence rules are applied in parallel. In the second step the adjunct rules are applied. In order to map the German dependency tree to phrase structures more complex rules are used. The context on the right hand side is frequently deeply embedded.

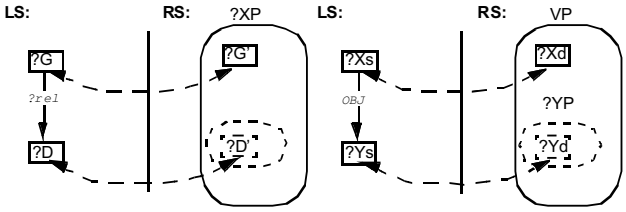

Figure 4: Domain adjunct rule pattern and example

Word order rules fall into two rule types: vertical rules and horizontal rules.

Vertical rules order a parent node in relation to a child node. The left-hand side of a rule consists of a path from the node p to the node c. In the most cases the path consists of only one edge. Figure 5 shows a rule pattern and example rule. The order of the nodes of the right-hand side is adapted depending on the order of the two nodes in the sentence of the training sentences. The left-hand side of the example rule matches the edges labelled as NMOD and the right-hand side orders the child node before the parent node. This rule is applicable to the nodes *book* and *a* and orders the *a* before *book*. The rule pattern shows several variants. In order to build topological models for German sentences, some of these rules have to access the right-hand side, for instance in order to determine whether a verb goes into the left or right bracket.

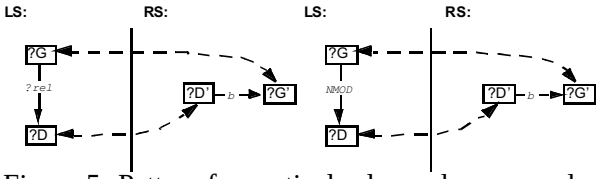

Figure 5: Pattern for vertical rules and an example.

Horizontal rules order two child nodes or grant children. The left-hand side of a rule consists of two paths starting at a node p and ending at the nodes v and w. Figure 6 shows a typical pattern of a horizontal rule. The example rule matches the nodes *read*, *book* and *about*.

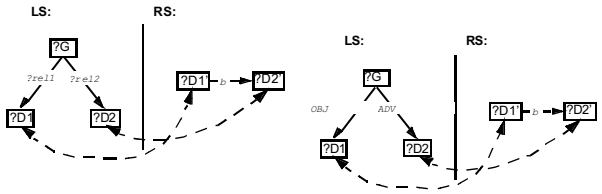

Figure 6: Rule pattern of horizontal rules and an example.

4 Induction of Linearization Grammars

In order to learn linearization grammars, we use the defined rule patterns to derive rules from pairs of dependency trees and topological models. The rules are build by completing the rule patterns. The left-hand side is completed with edge names and conditions which are taken from the dependency tree and the right-hand side is completed by labels as well as the direction of the nodes which have to be ordered.

For some of the rule patterns variations have to be build. For the domain adjunct rules, rule patterns with different embedding depth of the domain are examined until a possible solution is found.

The size of the left-hand side is increased in order to order all nodes of a domain. The advantage of this method is that it adapts automatically to different topoloical models which might have domains of different size.

Fundamentally for this method and the termination of the learning process is a termination condition which defines when the extension of rules is stopped. For the definition of the termination condition, we need to know which nodes have to be order by the rules. These are not only the nodes of a domain. They include also the children in the dependency tree of nodes contained in a domain. The children are required to order the domain in relation to other domains. We call this set the order set. Figure 7 shows on the left a dependency tree overlapped by the order set and on the right a dependency tree overlapped by domains.

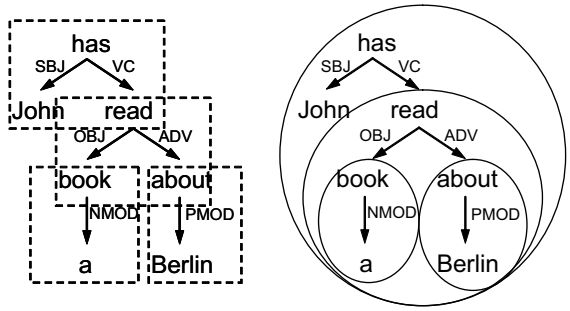

Figure 7: Dependency trees overlaped with oder sets (left) and domains (right)

Def. 3 *[order set]*
O_p is a set of nodes consisting of all nodes of a domain d_i and the target nodes of all dependency edges where the corresponding source node is contained in the domain:
Given that $G_T = \langle N_T, E_T \rangle$ is a dependency tree and $H_p = \langle G_P, G_D, G_c \rangle$ a precedence graph where the DAG $G_D = \langle N_D, E_D \rangle$ represents the domain hierarchy, $G_C = \langle W_c, D_c, E_c \rangle$ the coupling graph between the graph G_P and the domain hierarchy G_D, the function $f : N_T \rightarrow N_P$ maps the words of the dependency tree to the words of the precedence graph H_p, the set $D_p = \{s|(s,d) \in E_c\}$ contains the nodes included in a domain p, then the order set for the domain p is defined by the set of nodes: $O_p = \{\ n\ |(n = s \vee n = t) \wedge\ (s,t) \in E_T \wedge f(s) \in D_p \vee p \in D_c x\}$

With this definition, it is possible to define the *termination condition* for the rule creation in the learning algorithm: *Either the order set O_p it totally ordered* **or** *no further rules can be built ordering further elements.* The learning algorithm is shown below:

for all paris of the input structure $P_i = (G_i, H_I)$ **do**
  for all O_p^i of P_i do

    s ←1
    repeat

     build all rules for the rule patterns of size s

     if the rules were already built **then**
       increase the rule counter
     else store the rule

     s ← s + 1

    until O_p is totally ordered **or**
      no more rules can be built
      (cf. termination condition)

A grammar can cause cycles in a precedence graph. In order to solve this problem, we keep all rules and we assign a normalized count of the occurrences as weight to the edges. The cycles are dissolved in the precedence graph by removing one edge from each cycle with the lowest weight. The derived order is one with a high probability. This is sufficient in text generation and machine translation since we always look for the order, which fits best.

5 Evaluation of the Linearization Grammars

Due to the lack of an automatic evaluation method for the results of linearization components, we suggest and apply a method which computes the similarity between different word orders of a sentence. With this method, it is possible to measure the difference between the original word order and the derived word order. For languages with fixed word order, the value is correlated as shown below with the error rate of a sentence and in the case of languages with free word order, the evaluation method would probably work when the information structure would be annotated.

A measured difference does not always indicate an error since for instance prepositional constituents can be placed at different positions also within languages having fixed word order. In order to resolve this, it would be possible to define the exceptions manually. We think that this is acceptable since it is not possible to have an automatic evaluation method for all cases.

The similarity measure is defined between two word orders of a sentence based on the domains. The position of each elements of all domains from two orders are compared and the elements having the equal position are counted. The similarity is dfinied by the fraction of this count to the total number of the elements. In the following exapmle, the order of the two elements in domain 2 is switched. Therefore, the count of the elements taking the same positions in this domains is zero.

[John has [read [a book]$_1$]$_2$]$_3$
[John has [[a book]$_1$ read]$_2$]$_3$

The count of domain 1 is 2 and the count of domain 3 is 3. The number of correct placed

elements is 5 and the total number of elements is 7. The similarity is therefore 5/7 (0.714). The following equation defines the similarity relation:

$$\frac{\sum_{i=1}^{d} \texttt{elements with equal position in } D_i^{s1} \texttt{ and } D_i^{s2}}{\texttt{number of words} + \texttt{number of domains} - 1}$$

In order to evaluate the learning procedure for linearization grammars, we trained grammars on selected corpora as used in the shared task for dependency parsing for Catalan, English, German, and Italian. For the languages with fixed word order, we used the automatic evaluation method. For German, we had still to evaluate manually the results as in most of the cases, we did not get the original word order. One important reason for this is the missing information structure within the dependency trees.

As input for the training of the Catalan, Chinese, and English grammar, we used pairs of dependency trees and topological models that can be derive from the dependency trees. Each node was placed top down in a domain and only the leafs of the dependency tree together with their parent in one domain, cf. Figure 7 right.

The punctuation marks and coordinations are excluded from the training and evaluation since the annotation did not allow to derive the word order for these parts. The constituents of a coordination are all placed at the conjunction. Other annotation for dependency tree define a structure which would allow to derive the order, cf. (Mel'čuk and Pertsov, 1987).

Table 1 shows the results of the evaluation. The first column shows the language of the used corpus; the second column shows the number of sentences used for training; the third column shows the number of test sentences; the fourth column shows the average test sentence length; the fifth column shows the automatically computed similarity value between the original sentences and the sentences with the derived word order; the sixth column shows the average of the manual evaluation where results of test set has been rated with good, acceptable, and wrong. The value gives the accuracy for sentences rated with good or acceptable. The last value shows the average of the correlation between each of then ratings and each of the similarity values. In order to compute the correlation, the ratings has been mapped to numerical values. Since the sentence length is an important factor for the accuracy, we computed the correlation between the similarity values and length of the English sentences which is 0.22. The test sentences of the Catalan corpus are unusually long and for English a bit longer as the average of 24 words. The longer Catalan sentences are probably the reason for the slightly worser result compared to English.

Corpus (lang.)	Training (# sent.)	Test (# sent.)	Length (# words)	Similarity %	Eval.	Corr.
Catalan	14796	50	35.3	0.904	84%	0.53
English	18526	50	27.2	0.922	86%	0.51
Italian	3049	50	19.3	0.879	70%	0.42

Table 1: Summary of the results.

As input for German, we used the dependency tree and phrase structure annotation of the Tiger corpus. For German the usage of phrase structures as topological models, is not the best choice, since then not all possible word orders can be derived. But there are no large corpora for written text annotated with the German topological model. Additionally, we used about 10 hand written rules in order to assign additional features to the nodes that indicate the position of the verbs in the German topological model. The rule pattern had then also to consider this additional features during the learning phase.

Since German has a free word order, we evaluated 100 sentences manually, 21 of the sentences have been wrong, that is an accuracy of 79%. In 8 cases the pre-field have been empty that is caused by using phrase structures as topological models. In 5 cases rules have been missing and in 4 cases adverbs have been placed wrong. Each of the following cases occurred ones: wrong order of adjectives in a noun phrase, wrong order of a date format, wrong position of a verb due to the hand written rules, and finally an unspecified position of a negation particle. For German, we conducted another experiment and evaluated 20 randomly selected sentences with a length up to 12 words as sentences used in text generation are currently shorter and not so complex. We got only one error, which relates to an accuracy of 96%.

The tool for the induction of linearization grammars is embedded in a linguistic environment. We provide the environment including the tool for learning the linearization grammars and a description how to proceed.[5]

[5]Request are welcome and should be directed via email to the author of the paper.

6 Conclusion

We identified the two basic concepts to build topological models which underlay probably any topological model. The basic concepts allow to describe complex word order models, such as the classic German topological model or simpler models for languages with fixed word order. Based on this concpets rule patterns have been formulated using a tree/graph transducers formalism. The graph transducer use two tapes. Therefore, they have a lot of advantages. The most important one is that they allow to describe word order rules due to the syntactic necessities and to apply rules independently in parallel. The human brain is also massively parallel and is certainly also able to handle word order in parallel and not top-down as many algorithms do.

We introduce a method for the induction of word order rules, which takes as input sentences annotated with a dependency tree and topological model. The algorithm uses rule patterns to learn grammars. The results have been evaluated automatic for languages having fix word order. The learning and evaluation methods can be easily adapted to other languages.

References

G. Bech. 1955. *Studium über das deutsche Verbum infinitum*. Max Niemeyer Verlag, Tübingen.

B. Bohnet and H. Seniv. 2004. Mapping Dependency Structures to Phrase Structures and the Automatic Acquisition of Mapping Rules. In *LREC*, Portugal, Lisboa.

B. Bohnet. 2006. *Textgenerierung durch Transduktion linguistischer Strukturen.* Ph.D. thesis, University Stuttgart.

N. Bröker. 1997. *Eine Depedenzgrammatik zur Kopplung heterogeer Wissensysteme auf modalloischer Basis.* Ph.D. thesis, Albert-Ludwigs-Universitt, Freiburg.

G. Busatto. 2002. *An Abstract Model of Hierarchical Graphs and Hierarchical Graph Transformationbusatto*. Ph.D. thesis, Universität Paderborn.

Y. Ding and M. Palmer. 2005. Machine translation using probabilistic synchronous dependency insertion grammars. In *ACL '05*.

C. Donohue and I. Sag. 1999. Domains in Warlpiri. In *HPSG99*, Edinburgh.

E. Drach. 1937. *Grundgedanken der deutschen Satzlehre*. Diesterweg, Frankfurt.

D. Duchier and R. Debusmann. 2001. Topological dependency trees: A constraint-based account of linear precedence. In *Proceedings of the ACL*.

K. Gerdes. 2002. *Topologie et grammaires formelles de lallemand*. Ph.D. thesis, Universit Paris 7.

G. Gierz, K. H. Hofmann, K. Keimel, J. D. Lawson, M. Mislove, and D. S. Scott. 2003. *Continuous Lattices and Domains.*

T. Höhle. 1986. Der Begriff Mittelfeld. Anmerkungen über die Theorie der topologischen Felder. Akten des 7. Internationalen Germanistenkongresses, Band 3, pages 329–340. Tübingen.

A. Kathol and C. Pollard. 1995. Extraposition via complex domain formation. In *Meeting of the Association for Computational Linguistics*, pages 174–180.

K. Knight and J. Graehl. 2005. An overview of probabilistic tree transducers for natural language processing. In *Sixth International Conference on Intelligent Text Processing and Computational Linguistics*. Lecture Notes in Computer Science.

B. Lavoie, R. Kittredge, T. Korelsky, and O. Rambow. 2000. A Framework for MT and Multilingual NLG Systems Based on Uniform Lexico-Structural Processing. In *ANLP/NAACL Conference*.

C. Mellish, D. Scott, L. Cahill, D. Paiva, R. Evans, and M. Reape. 2006. A reference architecture for natural language generation systems. *Nat. Lang. Eng.*, 12(1).

I. A. Mel'čuk and N. Pertsov. 1987. *Surface-syntax of English, a formal model in the Meaning-Text Theory.* Benjamins, Amsterdam/Philadelphia.

I. Mel'čuk and L. Wanner. 2006. Syntactic mismatches in machine translation. *Machine Translation*, 20(2).

O. Rambow. 1994. *Formal and Computational Aspects of Natural Language Syntax*. Ph.D. thesis, University of Pennsylvania.

M. Reape. 1993. *A Formal Theory of Word Order. A Case Study in West Germanic.* Ph.D. thesis, University of Edinburg.

P. Skarup. 1975. Les premierès zones de la proposition en ancien francais. In *Akademisk Forlag*, Copenhaen.

M. Čmejrek, J. Cuřín, and J. Havelka. 2003. Czech-english dependency-based machine translation. In *EACL '03*.

F. Xia and M. Palmer. 2001. Converting Dependency Structures to Phrase Structures. In *The Proc. of the Human Language Technology Conference*, San Diego.