Unsupervised, graph-based learning of topological fields

Anonymous

Abstract

This paper presents an unsupervised approach to learning of symbolic word order rules from treebanks. We use a graph algorithm to compute topological field models, and employ decision tree learning to allow for robust automatic feature selection. The approach is applied to the German NEGRA treebank, and evaluated against the supervised approach of (Becker and Frank, 2002). We achieve 73.1% bracket recall, and discover phenomena as verb secondness and fine grained rules about middlefield order and scrambling.

1 Introduction

There is a simple reason for why we need to address the issue of how to learn word order rules: *Word order matters*. More and more treebanks are created for an ever-increasing number of languages. This gives us access to data with a broad range of word order variation in the structures we encounter. In §2 we discuss an empirical investigation that demonstrates there is significant variation in the amount of discontinuity and scrambling we can observe in treebanks for English, German, Dutch and Czech.

Hence, an approach for learning word order rules must be able to capture both *restrictions* on word order (e.g. V2-ness in German, Dutch) and *possible variability* in word order (e.g. scrambling in German, Czech). For that, we need rules – we cannot entirely lexicalize restrictions and possible variations:

First, restrictions like V2-ness hold for the broader context of a tree, not for individual lexical items. Second, particularly in languages with a freer word order, like German and Czech, word order helps realizing information structure. The latter is not lexicalizable because its interpretation is dependent on the actual discourse context.

We are not the first to discuss an approach to learning word order models. A related effort is (Becker and Frank, 2002), who present a supervised method for learning topological field models from the NEGRA treebank (Skut et al., 1997). Becker & Frank score high on chunking trees into separate fields (>91% LP/LR) but its supervised nature requires a hand-crafted gold standard annotated for the topological fields to be learned, and thus makes it costly to apply across many languages. In §5 we evaluate our approach against Becker & Frank's chunker, obtaining up to 73.1% bracket recall, and an average of 0.65 crossing brackets per sentence.

Few of the prominent context-free (CF) approaches like (Collins, 1997) have in fact been applied to other languages than English. (Collins et al., 1999) apply the parsing model of (Collins, 1997) to a fragment of the Prague Dependency Treebank (Hajič, 1998), and report 80% dependency accuracy. However, they do not describe the nature of the data, nor the learning behaviour. This is noted by (Dubey and Keller, 2003), who report 79.8% on parsing full NEGRA. It is therefore difficult to judge in how far the results of (Collins et al., 1999) can still be improved.

In this paper, we present a unsupervised, symbolic, and robust approach to learning word order rules, which is designed to be suitable for a wide variety of typologically different languages. The learning architecture consists of a *climbing module* working on dependency trees, a graph-based *field induction* component, and a decision-tree learning based

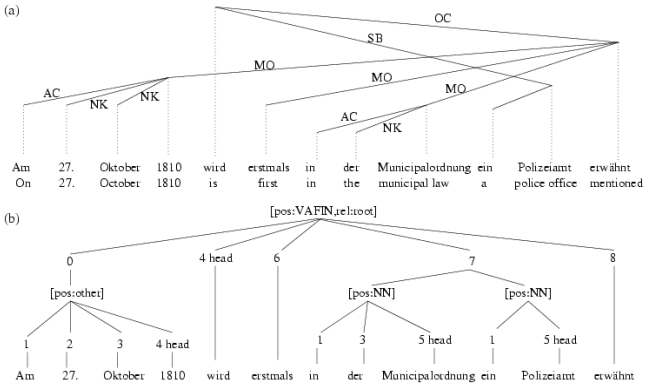


Figure 1: Dependency analysis of a sentence with a discontinous VP (NEGRA sentence 165). (a) TDG ID tree (b) induced and exported TDG LP tree

feature selection component. The task of the climbing component is to predict the behaviour of discontinous realizations. The task of the field induction component is to learn order rules for each class of governor found. Finally, the task of the feature selection component is to converge towards an optimal set of linguistic features to select as relevant to word order. The main advantages of the system are its minimal assumptions about linguistic structure, which makes it applicable to a wide range of languages, and its flexible feature selection component, which is capable of discovering very fine grained rules governing word order.

The paper is structured as follows. In §2, we describe properties of the linguistic data. In §3, we describe the learning architecture in detail, before we present its implementation in §4. Finally, we evaluate the system in §5.

2 Data

In this section we present an empirical treebank investigation into two aspects of word-order variation: discontinuity and scrambling. Both are difficult to deal with in context-free approaches. We used the WSJ section of the PennTreebank (Marcus and others, 1995) for English, the NEGRA treebank (Skut et al., 1997) for German, the Corpus Spoken Dutch (Oostdijk, 2000) for Dutch, and Row 5 from the Prague Dependency Treebank (Hajič, 1998) for Czech.

We use the following notion of discontinuity. Given a constituent (tree) that has a yield covering string positions $i...l$. The constituent is *discontinuous* if there is at least one interval of string positions $j...k$, with $i < j \leq k < l$, that does not belong to the yield of the constituent. Figure 1(a) shows a dependency tree with a VP (node "erwähnt") that is discontinuous in two places. We call an edge that is crossing a gap a *crossing edge*.

For each treebank, we investigate the discontinuity for nodes with an occurrence frequency $> 1\%$ relative to the total number of nodes observed in the treebank. Our coverage is at least 96.30% for each of the corpora. Table 1 presents an overview of the results. Σ^0 indicates the percentage of continuous nodes, and Σ^n the percentage of nodes discontinuous in n place(s); $\Sigma^d = \sum_{i=1..n}(\Sigma^i)$.

Language	Σ^0	Σ^1	Σ^2	Σ^3
English	96.38%	1.02%	0.01%	0.00%
German	86.33%	9.17%	1.82%	0.06%
Dutch	77.89%	16.44%	1.80%	0.20%
Czech	41.75%	49.46%	7.55%	1.06%

Table 1: Overview of continuity/discontinuity

Table 1 shows that the data confirms the expectation that a freer word order corresponds to a higher degree of discontinuity. The difference between German and Dutch arises from cross-serial dependencies in Dutch verbal clusters, which add $\Sigma^d = 4.12\%$.

Besides discontinuity, we also investigated scrambling. Both are normally associated with "free" word order. The difference between them is that discontinuity is a non-local phenomenon, whereas scrambling is head-local. To quantify scrambling, we use the following simple measure calculated on the basis of (ordered) dependency trees, and the (unordered) dependency mobiles they instantiate, all of the same type: *scrambling factor* $= log_2\left(\frac{|\text{unique dependency trees}|}{|\text{unique dependency mobiles}|}\right)$.

Table 2 shows an overview of the scrambling factors for different types of phrases, across the four languages. The closer to 0 a scrambling factor is, the less scrambling takes place.

The facts about verbal position and extraposition are traditionally analysed in terms of *topological fields* (Höhle, 1986). While topological field analyses have mainly been applied to Germanic lan-

Type	English	Dutch	German	Czech
S	0.31	1.20	1.02	0.56
VP	0.22	–	1.02	–

Table 2: Overview of scrambling factors

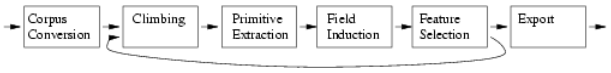


Figure 2: System Architecture

guages, a generalized version of this theory has proven to be useful cross-linguistically (Kruijff, 2001). Topological fields have also been successfully used in parsing.

3 Theory

We analyse sentences with a notion we borrow from Topological Dependency Grammar (TDG, (Duchier and Debusmann, 2001)): For a complete syntactic analysis we stipulate two tree structures, an ID structure describing syntactic functor-argument structure, and an LP structure describing linear precedence in terms of a generalized topological fields theory. Reconsider Figure 1 for illustration. ID trees are labelled with syntactic functions, and may have crossing edges. LP trees are labelled with topological fields, and are *projective*, i.e. may not have crossing edges. Nodes may *climb* to parent nodes when mapping from ID to LP trees. More formally, if w is the direct governor of v on the ID level and $w_1 \dots w_n$, are v's indirect governors, the governor of v on the LP level needs to be one of $w, w_1, \dots w_n$.

Given the above, the task of a machine learning system is to learn rules how to map from ID trees to maximally compact LP trees. The search space is clearly infinite. The learning task is modularized as depicted in Figure 2. In the remainder of this section, we outline each of the components.

3.1 Mathematical Model

The **Climbing Component** converts the input corpus into a dependency representation, and identifies crossing edges. It then produces projective versions of the input trees ("climb trees") by raising nodes until projectivity is reached. These trees are the input to the field induction component.

Primitive Extraction traverses all climb trees, and conceives of them as a set of primitive examples, from which we can learn word order regularities. Our primitive is a **precedence pair**, which we define as a single occurrence of a dependent item directly or indirectly before another dependent item under a common governor. We formalise a precedence pair as a quadruple $\langle e_1, e_2, R, h \rangle$, where e_1, e_2, h are feature structures (AVMs) describing dependency nodes, and R is equal to $<$ if e_1 preceeds e_2, and $>$ otherwise. The order of e_1 and e_2 in the quadruple is well-defined according to an arbitrary order on feature structures, e.g. the lexicographic order.

To map from dependency nodes to feature structures e_1, e_2, h, we define a **dependent feature selection function** $f : N \rightarrow E$ and a **head feature selection function** $f_H : N \rightarrow H$, where N is the set of nodes in the corpus, and $e_1, e_2 \in E, h \in H$ (We will refer to E as the set of *elements*, and to H as the set of *head classes*). Here, we assume f_H as given. We speak of *manual feature selection* when also f is known a priori, as opposed to *automatic feature selection*, when it is to be learned.

For illustration, assume that f_H maps nodes to feature structures containing the POS-tag of the node only, while f maps nodes to feature structures containing the syntactic relation of the node only. The node "wird" in Figure 1 will then be reduced to the AVM [pos:VAFIN]. "Oktober" will be a dependent of "wird" in the climb tree, since the edge has to climb in order to be non-crossing, and will be reduced to [rel:MO] by the dependent feature selection function, while "erwähnt" will be reduced to [rel:OC]. One of the precedence pairs of this local tree configuration is therefore $\langle$[rel:MO], [rel:OC], $<$, [pos:VAFIN]$\rangle$. Note that also the occurrence of the head itself among its dependents produces a precedence pair, e.g. the pair $\langle$[rel:HEAD], [rel:MO], $>$, [pos:VAFIN]$\rangle$.

We learn field descriptions from the set of precedence pairs observed in the corpus, with one field description for each head class $h \in H$ observed. Let $E_h \subset E$ be the set of elements occurring in all those precedence pairs that share the head h. We can then define the **field description** of head class h as a partial order on the dependent elements E_h.

To compute such a partial order, imagine a graph

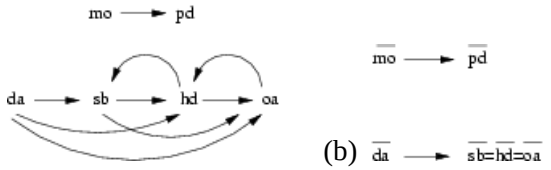


Figure 3: (a) Possible order graph for head class $h =$ "finite verbs". (b) Corresponding solution graph.

G, whose set of nodes corresponds to E_h. Iff there is a precedence pair $\langle e_1, e_2, <, h\rangle$, let there be an edge from the node in G corresponding to e_1 to the one corresponding to e_2. Let there be an edge in the opposite direction iff there is a precedence pair $\langle e_1, e_2, >, h\rangle$. We call such a graph an **order graph**. Figure 3(a) shows a possible order graph.

While a directed, acyclic graph (DAG) defines a partial order on its nodes, the order graph can obviously contain cycles. However, it is possible to identify the **strongly connected components** (s.c.c.)[1] of a graph, and reduce all nodes in the same s.c.c. to single nodes. (Mehlhorn, 1999) presents a linear-time algorithm for this task. We call the output of this algorithm a solution graph. A mapping from nodes of the solution graph to positive, consecutive integers can be interpreted as a **field description**. A field is then the set of nodes in the order graph mapped to the same integer. The solution graph in Figure 3(b) can be written as an assignment $\{mo, da\} \mapsto 0$, $\{sb, hd, oa, pd\} \mapsto 1$, and constitutes a field description with two fields.

3.2 Robustness and Feature Selection

Up to now we have assumed the dependent feature selection function as fixed. We now add a component that is able to adapt the feature selection function in a feedback loop with the field induction component, making the approach robust Figure 2.

Cycles in the order graph between elements a and b can be subclassified as follows. (a) there is no word order rule between a and b, their occurrence varies freely; (b) there is a word order rule between a subclass of a and a subclass of b, but it cannot be observed due to a wrong feature selection function.

We employ decision tree learning to adapt the dependent feature selection function. Decision tree learning has the advantage of being fast, applicable to many linguistic features, and producing symbolic rules as output, augmented with probabilities.

Before we outline an algorithm converging towards a near-optimal feature selection function, we introduce the frequency of a precedence pair, $c(p)$ as a measure of reliability of the observed precedence relation. We also introduce a **precedence table** as a convenient way to depict all precedence pairs observed under a common head. See Figure 4 for an example of a precedence table. The upper half of the symmetric matrix is mirrored to the bottom. Cell counts indicate the count of the precedence pair $\langle row, col, <, h\rangle$ and $\langle row, col, >, h\rangle$.

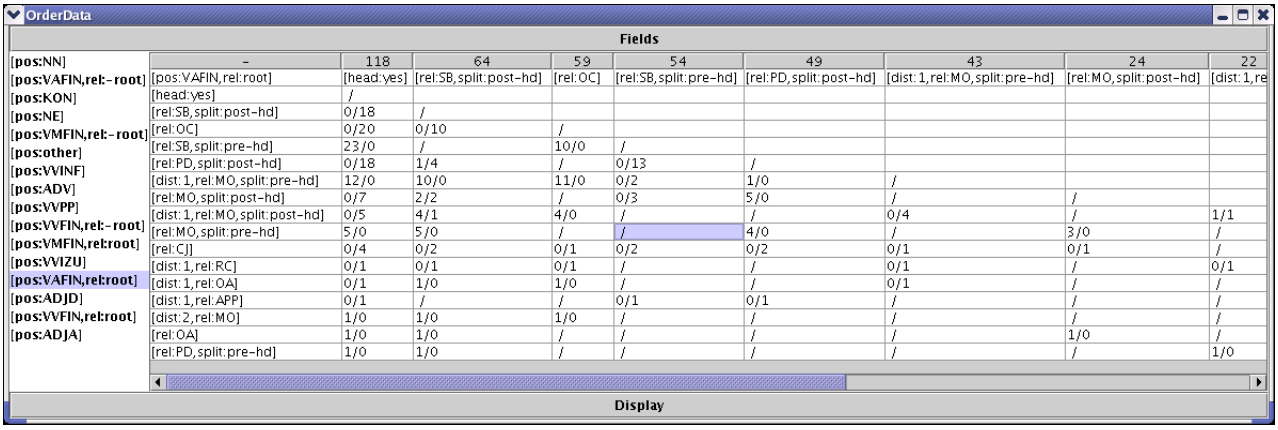
The feature adaption algorithm can be outlined as follows:

1. Start with a trivial feature selection function, e.g. one that distingushes head and dependent item only.
2. Pick a suitable precedence pair $\langle e_1, e_2, R, h\rangle$ from the precedence table
3. Invoke a decision tree learner on all precedence pairs, where input parameters are all features of e_1, e_2 and h, and R is the output variable. Interpret decision tree leaves as rules, and subclassify e_1 with all features of e_1 referred to in all rules found. Same for e_2.
4. While not converged, go to 2.

Some comments on this algorithm are necessary. When we pick a precedence pair in step 2, we choose the one with maximal **conflict ratio** $q(\langle e_1, e_2\rangle) = \frac{min(c_1,c_2)}{max(c_1,c_2)}$, $c_1 = c(\langle e_1, e_2, <, h\rangle)$, $c_2 = c(\langle e_1, e_2, > , h\rangle)$. At step 3, it should be pointed out that the classes found by the decision tree learner need not be disjoint, but that there is an order on them. The convergence criterion in step 4, finally, can be chosen as a system parameter.

The model as defined up to now is not able to discover head-secondness in the data, a criterion we characterized as essential in the data section of this paper. The reason is that elements that can occur either before or after the head will always be on the same strongly connected component as the head in the order graph. It is, in fact, nothing more than the linguistic intuition of head-dependent assymetry that justifies the head as a central element in a field description. We therefore introduce the **splitting rule**, which states: If decision tree learning fails on a precedence pair, one of whose elements is the head element, sub-classify the non-head element into two classes, "pre-head", and "post-head".

[1] A strongly connected component is a maximal subgraph in which any node can be reached from any other node



	-	118	64	59	54	49	43	24	22
[pos:NN]	[pos:VAFIN,rel:root]	[head:yes]	[rel:SB,split:post-hd]	[rel:OC]	[rel:SB,split:pre-hd]	[rel:PD,split:post-hd]	[dist:1,rel:MO,split:pre-hd]	[rel:MO,split:post-hd]	[dist:1,re
[pos:VAFIN,rel:-root]	[head:yes]	/							
[pos:KON]	[rel:SB,split:post-hd]	0/18	/						
[pos:NE]	[rel:OC]	0/20	0/10	/					
[pos:VMFIN,rel:-root]	[rel:SB,split:pre-hd]	23/0	/	10/0	/				
[pos:other]	[rel:PD,split:post-hd]	0/18	1/4	/	0/13	/			
[pos:VVINF]	[dist:1,rel:MO,split:pre-hd]	12/0	10/0	11/0	0/2	1/0	/		
[pos:ADV]	[rel:MO,split:post-hd]	0/7	2/2	/	0/3	5/0	/	/	
[pos:VVPP]	[dist:1,rel:MO,split:post-hd]	0/5	4/1	4/0	/	/	0/4	/	1/1
[pos:VVFIN,rel:-root]	[rel:MO,split:pre-hd]	5/0	5/0	/	/	4/0	/	3/0	/
[pos:VMFIN,rel:root]	[rel:CJ]	0/4	0/2	0/1	0/2	0/2	0/1	0/1	/
[pos:VVIZU]	[dist:1,rel:RC]	0/1	0/1	0/1	/	/	0/1	/	0/1
[pos:VAFIN,rel:root]	[dist:1,rel:OA]	0/1	1/0	1/0	/	/	0/1	/	/
[pos:ADJD]	[dist:1,rel:APP]	0/1	/	/	0/1	0/1	/	/	/
[pos:VVFIN,rel:root]	[dist:2,rel:MO]	1/0	1/0	1/0	/	/	/	/	/
[pos:ADJA]	[rel:OA]	1/0	1/0	/	/	/	/	1/0	/
	[rel:PD,split:pre-hd]	1/0	1/0	/	/	/	/	/	1/0

Figure 4: Precedence Table for finite auxiliary verbs at the root node, calculated from 180 sentences of Negra, with options +climb, +split, and a manual feature selection strategy for syntactic relation

4 Implementation

The architecture outlined above is fully implemented in Java. The system accepts any corpus in the widely-spread Negra export format, and features some 20 options to influence system behaviour. The C4.5 decision tree learner (Quinlan, 1998) was used as an external tool for decision tree learning, and is automatically invoked from within the system. A run over 1000 sentences takes between 20 minutes and 2 hours, while a run on the entire corpus takes about a day. The system has an interactive mode with graphical data browsing facilities, and a batch mode.

The most important system options include: **climbing**, **splitting** and **feature adaption** by decision tree learning can be switched on or off, different manual feature selection strategies can be used; **threshold** values can be set that determine in what case an edge should be included into the order graph, a conflicting precedence pair should be transmitted to the decision tree learner, and what rule reliability is necessary to add the decision rule to the feature adaption function; and different **convergence** strategies can be used.

Figure 4 presents a screenshot of a precedence table, calculated from a small corpus to increase legability. Figure 5 presents the corresponding field description learned from this precedence table. Nodes of the order graph are aligned such that columns can be read as fields. *split* and *dist* (distance) features indicate split and climbed elements respectively.

The example resembles a topological field representation very closely, which is partly due to the split rule. Fields 0 and 1 constitute the verbal *vorfeld*, and contain elements like (climbed) modifiers ([rel:MO,split:pre-hd]), like “Am 27. Oktober” in sentence 165 above (Figure 1). The feature structure [rel:OC] in field 5 subsumes clausal objects, and thus constitutes the right sentence bracket. While the order of subjects and modifiers is predicted to be free (hence their occurrence in fields 0, 1 and 3), the order of non-local relative clauses is correctly restricted to a position to the right of the right sentence bracket (field 5). Field number 1 deviates from a standard topological model, but is justified by postnominal focus-particles (*The originality of the music, though, got lost ...*, Negra sentence 75).

5 Evaluation

The quality of the learned word order rules cannot be directly evaluated, apart from human judgement of linguistic adequateness. We therefore evaluate the system in two ways. First, we describe convergence behaviour and system parameters. Second, we evaluate against the results of (Becker and Frank, 2002).

5.1 System Behaviour

In this section we describe a set of experiments on 900 sentences, investigating the size of precedence tables and field descriptions. We controlled corpus size and system options to observe the convergence behaviour. We tested the following models, with these options:

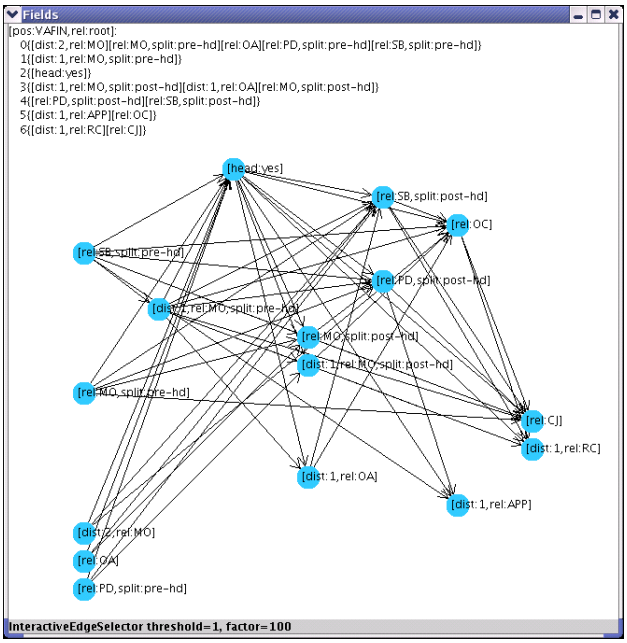


Figure 5: Field Description in graph format corresponding to the data in Figure 4

model	climb	adapt	split
full-adt10	y	auto	y
manual	y	manual	y
noClimbManual	n	manual	y
noAdapt	y	manual	n

In the diagrams in Figure 6, we plot the number of elements in the head's precedence table (left) and the number of fields in the learned field description of the head at the y-axis (right) for different models. A snapshot of the system is plotted after each corpus subsection of 100 sentences, indicated at the x-axis.

Precedence table size convergence depends primarily on the model used. While it is trivial for the *noAdapt* model, we achieve fast convergence within 900 sentences for the model *manual*, although non-local elements are a possible source of explosion, and the feature selection function for climbed elements matters. Precedence table size convergence for models with automatic feature adaption is more critical, and depends on the treshold values used (The model displayed stopped calling the decision tree learner as soon as precedence pair count was below 10). Figure 6 also shows that some tables converge slower than others, particularly the ones with a high amount of non-local elements (verbs).

The number of fields computed depends on the size of the underlying precedence tables, and on

[head:yes]	735	[rel:MO]	76
[pos:NN,rel:MO,split:pre-hd]	258	[split:pre-hd]	74
[rel:SB,split:pre-hd]	245	[rel:SVP]	68
[pos:ADV,rel:MO,split:pre-hd]	202	[pos:ADV,rel:MO,split:post-hd]	67
[pos:PRELS]	195	[pos:ADJD,rel:MO,split:pre-hd]	64
[rel:CP]	185	[rel:SB,split:post-hd]	60
[pos:NN,rel:OA,split:pre-hd]	153	[rel:OC,split:post-hd]	52
[pos:PPER,rel:SB,split:pre-hd]	141	[pos:ADJD,rel:MO,split:post-hd]	33
[pos:NN,rel:MO,split:post-hd]	131	[pos:PRF,rel:OA,split:post-hd]	32
[pos:NN,rel:OA,split:post-hd]	95	[rel:OC,split:pre-hd]	31
[pos:PRF,rel:OA,split:pre-hd]	85	[pos:ADJA,rel:MO,split:pre-hd]	30

Table 3: most frequent of classes under [pos:VVFIN,rel:-root] by automatic feature adaption

threshold values for introducing edges into the order graph. The field size for the model with automatic feature selection converges slower. While overall field size increases slowly, the curves are volatile for both models, a behaviour which is due to classes being collapsed into equivalence classes or established as fields, as soon as thresholds are exceeded.

To give a closer insight into the classes the decision tree learner picks as order relevant, Table 3 lists the most frequent elements in the precedence table of finite full verbs, computed by model *full-adt10*. The table includes linguistically fine-grained tendencies (Kurz, 2000) like subject pronominalization ([pos:PPER,rel:SB]). While most classes are split, there are also clearcut cases in the data, for instance separable verb affixes, which are consistently at the end ([pos:SVP]).

5.2 Comparison with Topological Corpus

For this experiment, we divided the input corpus into 9/10 training data and 1/10 test data, and used the learned word order rules to annotate the test data sentences with the fields learned during training. We exported the resulting trees (TDG LP trees), and compared the tree structures to the output of (Becker and Frank, 2002)'s parser, using the standard PARSEVAL measure, and the same tool (EVALB) as Becker and Frank to calculate unlabelled recall and the number of crossing brackets.

We focus on recall rather than precision or f-value, because a peculiarity of Becker and Frank's data is their virtually non-existent annotation of noun-phrase topology. Our system, however produces fine-grained NP-word order rules – a feature we do not want to be penalized for. We use unlabelled bracket recall rather than labelled, because Becker and Frank use linguistically meaningful topological field labels like *vf*, *mf* for *vorfeld*,

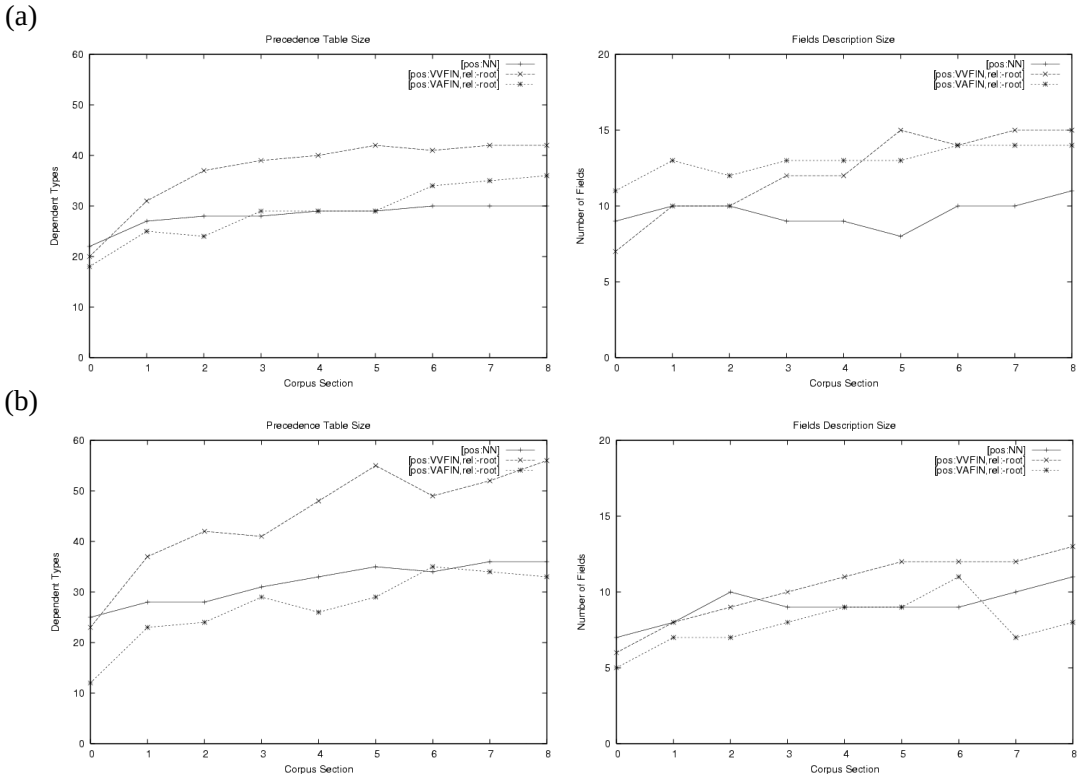


Figure 6: Precedence Table and Field Description growth on 900 sentences for (a) manual (b) full-adt10

middle field, etc., while the output of our system is numerical. We ignored brackets of span 1 as well as brackets that spanned the entire sentences.

Figure 7(a) shows that the system produces stable results on little data already. We get values of up to 73.1% unlabelled recall, and 95.0% sentences with 2 or less crossing brackets. This behaviour is similar for all models with splitting. The more fine grained analysis which automatic feature adaption provides does not improve the recall on the data with which we compare.

While we showed that bracket measures are independent of corpus size already for some hundred sentences, the measures depend strongly on the parameters of the model. Figure 7 shows that the splitting rule is the most important parameter that determines recall figures.

5.3 Discussion

The results show that the system produces fine grained results on little data already. Precedence table and field size converge quickly for manual feature adaption, while convergence is dependent on threshold values for automatic feature adaption. Since decision tree learning is a greedy strategy that produces only near-optimal results, further experiments with threshold values and significance testing of the discovered classes may improve the results.

The results showed that the splitting rule is a necessary adaption to the system in order to get sensible results. While 73.1% bracket recall leaves room for improvements, the majority of mismatching cases between the learned rules and the data by Becker and Frank we were using is due to linguistically different, but both plausible analysis. Our system consistently assumes an own field for verbal infinitive markers, and attaches relative clauses and arguments of non-finite verb forms low rather than high, contrary to the analysis of Becker and Frank. Also the analyses of conjunctions differ, but are both consistent. On the other side, our rules are more fine-grained, which is an advantage of our system that is not reflected in the bracket measures. It should also be stressed that our approach is unsupervised, and that supervised approaches tend to outperform the former.

(a) (b)

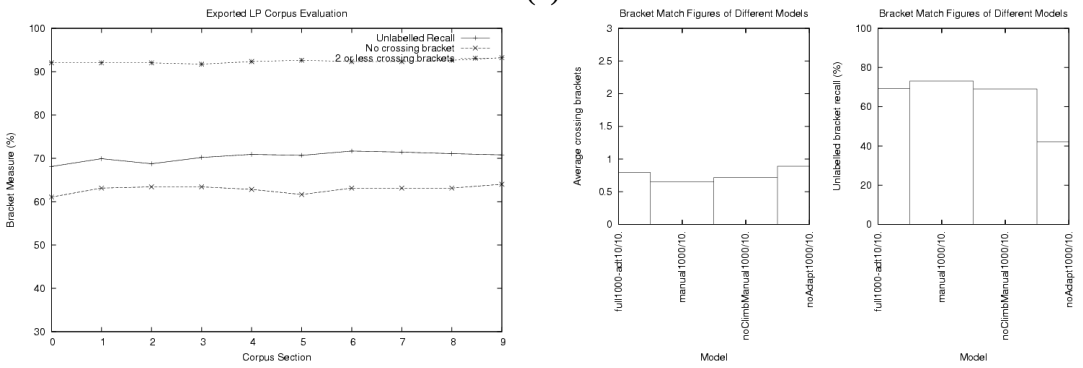


Figure 7: Influence of (a) Corpus Size (6000 training, 600 test) and (b) model on bracket measures

6 Conclusion

We gave empirical evidence that word order rules are a necessary component on the way towards automatic acquisition of syntactic knowledge. We then presented an unsupervised machine learning approach that robustly learns symbolic word order rules from treebanks, showing that topological fields are a useful means to describe word order regularities. The approach makes little assumptions about linguistic structures, contrasting e.g. to (Villavicencio, 2000)'s system, and is therefore more easily applicable to a wide range of typologically different languages. The approach is also able to learn a set of word-order relevant linguistic features. This shows that decision tree learning is not only applicable to a restricted word order phenomenon (Gamon et al., 2002), but can guide an entire machine learning system. We hope to improve evaluation figures in future research by including different climbing strategies, and adapting the convergence behaviour of the decision tree learner.

References

Markus Becker and Anette Frank. 2002. A stochastic topological parser for German. In *Proceedings COLING'02*.

Michael Collins, Jan Hajič, Lance Ramshaw, and Christoph Tillmann. 1999. A statistical parser for Czech. In *Proceedings ACL'99*.

Michael Collins. 1997. Three generative models for statistical parsing. In *Proceedings ACL'97*.

Amid Dubey and Frank Keller. 2003. Probabilistic parsing for German using sister-head dependencies. Under submission.

Denys Duchier and Ralph Debusmann. 2001. Topological dependency trees: A constraint-based account of linear precedence. In *Proceedings ACL'01*.

Michael Gamon, Eric Ringger, Zhu Hang, Robert Moore, and Simon Corston-Oliver. 2002. Extraposition: A case study in German sentence realization. In *Proc. COLING'02*.

Jan Hajič. 1998. Building a syntactically annotated corpus: The Prague Dependency Treebank. In Eva Hajičová, editor, *Issues of Valency and Meaning*, pages 106–132. Karolinum, Prague, Czech Republic.

Tilman Höhle. 1986. Der Begriff Mittelfeld: Anmerkungen über die Theorie der topologischen Felder. In Albrecht Schöhne, editor, *Germanisten-Kongress Göttingen 1985*, pages 329–340. Tübingen: Niemeyer.

Geert-Jan Kruijff. 2001. *A categorial-modal logical architecture of informativity*. Ph.D. thesis, Charles University, Prague.

Daniela Kurz. 2000. Wortstellungspräferenzen im Deutschen. Master's thesis, Saarland University.

Mitchell Marcus et al. 1995. The Ultimate Penn Treebank Bible. Technical Report CD, Linguistic Data Consortium (UPenn). Technical Report.

Kurt Mehlhorn. 1999. Graph Algorithms and NP-Completeness. www.mpi-sb.mpg.de/~mehlhorn/DatAlgbooks.html.

Nelleke Oostdijk. 2000. The Spoken Dutch Corpus. Overview and first evaluation. In *Proceedings LREC 2000*.

John Ross Quinlan. 1998. *C4.5: Programs for Machine Learning*. Kaufmann.

Wojciech Skut, Brigitte Krenn, Thorsten Brants, and Hans Uszkoreit. 1997. An annotation scheme for free word order languages. In *ANLP'97*.

Aline Villavicencio. 2000. The acquisition of word order by a computational learning system. In *Proceedings CoNLL*.