Structural Prediction in Incremental Dependency Parsing

Niels Beuck and Wolfgang Menzel

University of Hamburg
{beuck,menzel}@informatik.uni-hamburg.de

Abstract. For dependency structures of incomplete sentences to be fully connected, nodes in addition to those corresponding to the words in the sentence prefix are necessary. We analyze a German dependency corpus to estimate the extent of such predictive structures. We also present an incremental parser based on the WCDG framework and describe how the results from the corpus study can be used to adapt an existing weighted constraint dependency grammar for complete sentences to the case of incremental parsing.

Keywords: Parsing, Incrementality, Weighted Constraints, Dependency Grammar

1 Motivation

A dependency structure of a sentence is a fully connected, single headed, directed acyclic graph, where every node corresponds to a word in the given sentence. For incomplete sentences, as encountered during incremental parsing, achieving connectedness is often not possible when the dependency structure is limited to the nodes corresponding to words in the sentence prefix. This problem occurs for every word that is attached to the right, as there will be at least one sentence prefix where the word but not its head is part of the prefix.

There are two possible approaches to deal with this problem: Either to drop the connectedness requirement for the dependency structures of incomplete sentences (from now on called partial dependency analyses, PDA) or allow additional nodes in the structure as needed to establish connectedness. In the first variant, words are left unattached until a suitable head appears. While this approach to incremental dependency parsing has been successfully used in parsers like MaltParser [1], we argue that it is unsatisfactory when partial dependency structures are of interest, for several reasons:

From a psycholinguistic point of view: Humans not only have interpretations of partial sentences, but also have strong expectations about the integration of words even if their syntactic head is still missing [2].

From an application point of view: Information already contained in the perceived part of the sentence is missing in the dependency structure if words are left unconnected. In the incomplete sentence “Pick up the red”,

the head of 'red' is still missing, but we can already argue that the object of 'pick' is something red. This is not limited to words close to the end of the prefix, like in a noun phrase under construction. It also applies to long distance dependencies like in German subclauses, where the verb comes last and all its arguments would need to stay unconnected. [3] proposed explicit dependencies between the arguments, but these are unnecessary if we provide an extra node for the expected verb, allowing early attachment of the arguments.

From a parsing perspective: Parsers choose between alternative structures based on some measure of grammaticality or probability. The WCDG parser used in this work uses a set of weighted constraints to measure the grammaticality of dependency structures. To be able to reuse measures optimized for judging analyses of complete sentences, a structure similar to the final structure is required to provide a rating similar to the rating for the final structure. Otherwise, unintuitive structures with no possible counterpart in a complete sentence analysis might be rated unreasonably high and lead the parser astray.

On the one hand, additional nodes in partial dependency analyses are desirable because they allow more informative structures. This leaves open the question of how to construct such structures in a parser. On the other hand, a constraint based parser would penalize partial dependency structures even if the constraint violation could and probably will be remedied by further input, making it hard to fairly rate and select a suitable dependency structure for a given sentence prefix.

We propose that the second problem holds the answer to the first one. The very constraint violations that unnecessarily penalize partial dependency analyses lacking structural prediction can be used as indicators for how to extend these structures with the needed additional nodes.

The remainder of this paper consists of two parts: First, we will present a constraint based incremental parsing algorithm able to make structural predictions about the upcoming input. It uses an existing weighted constraint dependency grammar and an existing transformation based parsing algorithm as well as a set of potential placeholders, so called virtual nodes (VNs), for structural prediction.

Secondly, we will analyze a given corpus of dependency annotations to assess the additional structure needed to keep PDAs connected and to fulfill all valency requirements. From this result we can infer the set of VNs we have to provide to our algorithm to achieve a certain coverage. In addition, we analyze to what extend the given set of weighted constraints can be used to judge such structurally predictive partial analyses and where it has to be adapted, e.g. by adding suitable exceptions.

2 Related Work

While to our knowledge this is the first strictly incremental dependency parser (strict in the sense of providing a fully connected structure for every input in-

crement), strict incrementality has been explored already for the Tree Adjoining Grammar (TAG) formalism. [4] investigated the amount of structure needed for strictly incremental parsing by building gold standard annotations incrementally with a simulated parser. They introduced the term *connection path* to denote the set of nodes of the (connected) tree for increment n that are not part of the tree for increment $n-1$ and not part of the subtree provided by the new word. While they worked with phrase structures for English and only regarded predictions needed to achieve connectedness, our work is on dependency structures for German and regards valency requirements in addition to connectedness.

PLTAG (PsychoLinguistically motivated Tree Adjoining Grammar)[5] is a TAG variant, that allows strictly incremental parsing. There, connectedness is facilitated by so called prediction trees, non-lexicalized elementary trees whose nodes later need to be instantiated by nodes from lexicalized elementary trees. Prediction trees are not anchored in the input directly, but are integrated when necessary to connect the elementary tree of the new material to the structure built so far. As the number of possible connection paths grows polynomially with the number of prediction trees contained, the parser presented in [5] restricts connection paths to one prediction tree.

The PLTAG parser deals with temporary ambiguity by keeping a beam of possible derivation trees, ranked by their probability learned from a training corpus. Thus, the best structure needs not be a monotonic continuation of the previously best ranked structure. The output of the incremental WCDG parser is non-monotonic, too, but instead of working with a beam of monotonically extended structures, it works by successively transforming a single structure. In contrast to the PLTAG parser, incremental WCDG achieves state of the art accuracy (see [5], p.230 and [6]).

3 The WCDG Framework

Weighted Constraint Dependency Grammar (WCDG) [7] is a framework for dependency parsing by means of constraint optimization. The grammar consists of a set of constraints regarding single or pairs of dependency edges. So called context sensitive predicates allow constraints to access properties of the complete tree, e.g. to express the existence requirement for the subject of a verb. The constraints are weighted with a penalty score between 0 and 1, where 0 denotes a hard constraint. By applying all constraints of the grammar to a dependency structure, a grammaticality score for the structure can be calculated by multiplying the penalties of all constraint violations. Thus, parsing in WCDG is the search for the highest scored dependency structure. Every word can, in principle, be attached to any of the other words in the sentence, including the generic root node via any dependency type. The only restriction is that a word can only be attached to one head.[1] All further restrictions, like projectivity, are

[1] WCDG supports multiple levels, i.e. building multiple dependency structures at once, including constraints mediating between them. Yet, for a given level, a word can have only one head.

expressed via constraints. The number of possible edges grows quadratic with the number words in the sentence, resulting in an exponential number of possible dependency structures. Thus, for non-trivial sentences, a complete search is usually not feasible.

[7] presented a repair based heuristic algorithm called frobbing. The algorithm starts with an arbitrary initial structure and identifies candidates for replacement by looking at the constraint violation with the highest penalty. Replacement variants for the edges violating the selected constraint are then evaluated for whether they would prevent this conflict and one of them is applied. Several of these repair steps might have to be chained to escape from a local score maximum.

This algorithm is inherently non-incremental, as it starts with a structure for the whole sentence. An incremental processing mode can be introduced by applying the procedure on prefixes of the input and using the resulting dependency structure as the initial structure for analyzing the extended prefix. In this incremental mode, edges selected in a previous step can be replaced later on, if necessary. This can happen when, given the additional input, they violate additional constraints or when an old constraint violation can be resolved given the new input. The algorithm guarantees no monotonicity at all, but in practice an attachment stability of around 70% has been observed [6].

4 Structural Prediction with Virtual Nodes

When parsing a sentence prefix instead of a full sentence with WCDG, several constraints might prevent the parser from choosing a prefix of the structure which would be selected for the full sentence, i.e. a structure only containing dependency edges also present in the complete structure. This is foremost due to 'fragmentation' constraints, penalizing unattached words. They force the parser to select an unsuitable attachment point that might be penalized by other constraints, but not as hard as the fragmented reading. Also, valency constraints are commonly violated in such prefix structures.

There are two possible approaches to deal with these kinds of constraints when applied to partial dependency analyses: Either we prevent them from being applied to PDAs or we add as many additional nodes and edges to the dependency structure as needed to prevent the constraint violation. The first approach has the severe drawback of over-generation. Disabling constraints (or adding exceptions to them) will remove their ability to penalize ungrammatical input.

In this paper we follow the second approach by providing a set of so called virtual nodes that can be integrated into the dependency structure if a constraint violation can be avoided by doing so, but there is no penalty if they stay unconnected. If being unconnected, they are not considered to be part of the structure.

To prevent the parser from choosing a structure that includes virtual nodes even if another structure of equal or slightly worse score would be available, we

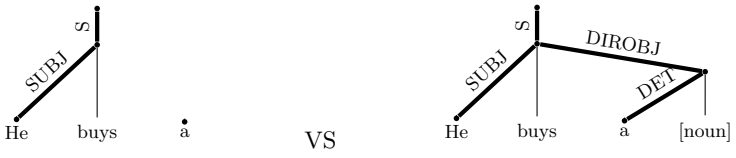

Fig. 1. Example for bottom-up prediction: The determiner is unattachable to any word directly, but attachable if predicting a noun

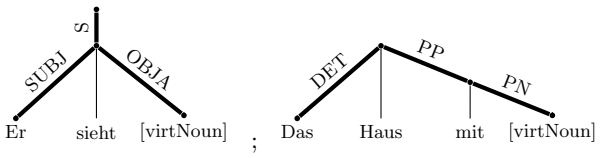

Fig. 2. Examples for top-down predictions of an object and the kernel noun of a preposition ("he sees" and "the house with")

add a slight penalty to every analysis where a virtual node is used, i.e. connected. This way, the structurally predictive reading is chosen only if necessary, but not otherwise. No further modification to the algorithm is needed. During constraint optimization the virtual nodes are treated like other input nodes. The special handling of virtual nodes (i.e., the slight penalty for connected and no penalty for unconnected virtual nodes, in contrast to the harsh penalty for other unconnected nodes) is defined solely in the constraints of the WCD-Grammar.

We can distinguish two different scenarios leading to structurally predictive readings. The scenario seen in Figure 1 is a bottom-up prediction, where the necessity to connect a new word leads to the prediction of a suitable head.

As seen in Figure 2, dependents can be predicted, too, via top down prediction. Whenever a word has a valency that can not be satisfied by any of the other available word, e.g. missing subject or a transitive verb missing an object, filling the valency with a VN makes the corresponding constraint violation disappear.

If we would allow some words to stay unattached, it cannot be guaranteed that there is any possible continuation of the sentence, where these unattached words could be attached without changing the already established part of the structure. For fully connected structurally predictive structures, in contrast, the absence of severe constraint violations licenses not only the integration of the otherwise unattachable words but also the acceptability of the whole structure. It anticipates a potential continuation of the sentence prefix, which does not introduce any additional constraint violations. The structural prediction serves as a kind of "certification" for the grammaticality of all the attachments in the PDA.

It has to be assured that for every sentence prefix the grammar licenses a partial dependency analysis free of additional constraint violations. Besides providing a suitable set of virtual nodes, grammar modifications might be necessary to achieve this goal, as will be discussed in Section 6.

To be able to predict structural relationships for yet unseen words, the incremental frobbing algorithm needs to be provided with a finite set of virtual nodes. The larger this set, the higher the coverage will be. Additional VNs, however, introduce a quadratic increase of potential dependency edges into the search space and, as a consequence, an exponential increase in the number of possible dependency structures.

For the sake of computational feasibility, the number of VNs should be kept as small as possible. This can be achieved by using general VNs (which stand for whole classes of words such as nouns or verbs). Aside from the combinatoric problems of providing too many VNs, it also avoids the danger of many variants receiving the same grammaticality score and thus being indistinguishable by the parser. Even the order among VNs will remain unspecified. While the existing grammar is prepared to deal with underspecified values for many lexical features, attributes like the word form or the linear precedence relations among words are always considered being given explicitly. To allow constraints to ignore the arbitrary order/adjacency relations between VNs as well as their particular lexical instantiations, respective exceptions need to be added.

Thus, the questions we have to answer are:

- Which set of virtual nodes is needed to achieve a certain coverage
- What changes to a constraint dependency grammar for full sentences are necessary?

In the second half of the paper we will present an empirical investigation to answer these questions.

5 Data Preparation

Not only we have no annotations for sentence prefixes available, but also it would be difficult to specify them uniquely, since they always depend on a hypothesis for a possible sentence continuation. Therefore, we generate them from complete sentence annotations. We want to find a monotonic sequence of prefixes resulting in the final annotation. Those prefixes are therefore not necessarily the most plausible for any given prefix.

For every word in a given sentence, we generate an annotation up to that word by retaining certain nodes and dependency edges while dismissing others[2].

5.1 Prefix Baseline Annotation

The basic variant of the procedure that generates connected PDAs is quite straightforward:

- all words inside the prefix and dependencies between them are kept
- all words that are heads of retained words are retained themselves, including the dependency edge between the head and the dependent. This rule applies recursively.

[2] The software to reproduce this data preparation as well as the evaluation can be found at http://www.CICLing.org/2013/data/274

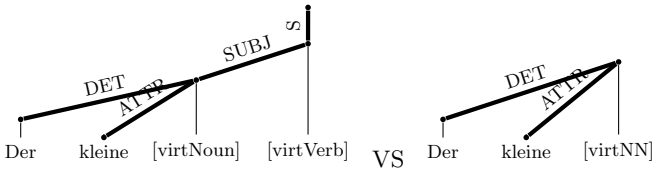

Fig. 3. Example for assuming a complete sentence vs a fragment given the prefix "der kleine" ("the little")

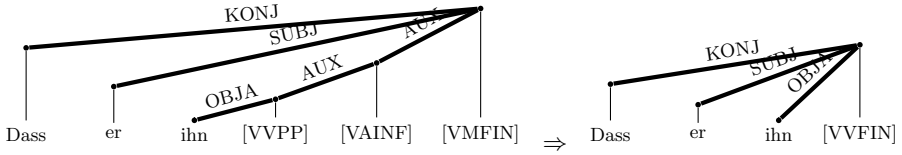

Fig. 4. Example for folding: When generating a PDA for the first three words of the clause "dass er ihn gesehen haben soll" (literally: "that he him saw has should" meaning:"that he is supposed to have seen him") . All three verbs would be kept for connectedness, but are then folded into a single virtual verb

Words not belonging to the prefix but nonetheless retained need to be 'virtualized' to produce generic representatives of their part-of-speech category, the form of the virtual nodes used by the incremental parsing algorithm.

The procedure of virtualization removes word form and lemma, keeps part-of-speech information, removes all other lexical features, and blurs the original linear order among the nodes.

5.2 Prefix Annotation Variants

Three parameters for the prefix generation algorithm have been considered:

1. Assuming complete sentences or not, i.e. whether we retain all nodes needed to connect up to the root or only retain those needed to connect all words in the prefix to each other (see Figure 3).
2. Top-down prediction. The connectedness criterion only provides for bottom-up prediction. Additional nodes could be retained via top-down prediction, i.e. to fill obligatory valencies (see Figure 2).
3. Folding of connection paths. The connection paths extracted directly from the full sentence annotations might not represent the shortest possible connection path. This is especially true for chains of equally labeled dependencies which correspond to nested phrase structures. Folding these structures by only keeping one of the dependencies will lead to more plausible prefix annotations (see Figure 4). They correspond to the minimally recursive structures [4] investigated in their second experiment where they anticipated parser miss-analyses in their simulated parsing procedure.

These three parameters have different consequences for the parser. Unless there are clear hints for a nested structure, the minimal tree, i.e. the one with the least number of virtual nodes, should always be preferred. WCDG achieves this

due to the prediction penalty, selecting the PDA with a minimal amount of VNs among otherwise equally scored alternatives. The parser will not be able to utilize additional VNs set aside for additional nesting depth. Folding can be expected to provide a better estimation of the amount of VNs needed.

While an incremental dependency parsing algorithm that provides bottom-up but no top-down prediction is possible, the parsing algorithm used here has no distinct mechanisms for the two prediction modes. If respective virtual nodes are provided, they will be used to satisfy valency constraints, unless those constraints are disabled for incomplete input. Therefore, by adding top-down prediction to the prefix generation algorithm, we get a better estimation for the VNs needed by the parser. Note that folding reduces the number of VNs, while top-down prediction increases it.

In contrast to folding, where there is no choice for parser behavior, the parser can easily be changed between a full sentence and a fragment mode. Which mode is activated depends solely on whether there is a constraint for penalizing words other than finite verbs to be the root of the tree. Such a constraint would incite the prediction of a virtual finite verb (and possibly its subject) even if none is needed to establish connectedness. Thus, the choice between these modes affects the number of required virtual nodes. In this paper we only explored the fragment variant, as did [4] and [5]. The predictions in this mode are expected to be smaller and less speculative.

6 Experiments

To determine the set of virtual nodes needed, we generate a partial but connected structure for every prefix of every sentence in an annotated corpus. A subset of the dependency edges of the complete tree is selected by using the procedure presented above. Given these structures, we can count how many additional nodes are needed to keep them connected. Also, by evaluating these PDAs with the WCD-grammar, we can determine whether they violate any constraints in addition to those violated in the complete sentence, as such additional constraint violations would prevent the prefix structure to be selected by the parser. Based on these results we can then modify the grammar accordingly.

Using the 500 sentences from the Negra corpus [8] transformed to the dependency annotation scheme described in [9] and a broad coverage WCD-grammar for German [7] we perform the following procedure:

1. generate a set of prefix annotations from the gold standard annotation.
2. Calculate constraint violations that only appear on prefix analyses, but not in the corresponding complete sentence
3. check:
   (a) whether changes to the constraints themselves are needed
   (b) whether additional VNs corresponding to words in the complete sentence could avoid any of the additional constraint violations
   (c) whether VNs can be removed from the connection path systematically without triggering additional constraint violations

4. repeat from step 1 with different variants of the PDA generation algorithm

6.1 Prediction with Baseline Annotations

We start with the most simple variant of the prefix generation algorithm, i.e. without folding or top-down prediction. The constraint violations encountered in the prefix annotations but not in the respective complete sentence annotation can be roughly categorized as follows:

Prediction penalties As described in Section 4, there is a small penalty for integrating VNs into a structure. The respective constraint violations are expected in the difference lists, and can safely be ignored here.

Order constraints, especially projectivity These constraint violations occur due to VNs being unordered. These constraints should simply not be applied to VNs. Hence, respective exceptions need to be added to them.

Unsatisfied valencies As we did not account for valencies when deciding on what to keep in the prefix structures of this iteration, all kinds of valency constraints are violated, e.g. missing subject, missing object, missing kernel noun for prepositions, missing determiners for nouns, missing conjunctions and missing conjuncts. These are prime examples for top-down prediction, i.e. these conflicts will disappear when attaching a VN with a respective PoS and dependency type. For VNs missing determiners or infinitive markers 'zu', PoS variants without these valencies (NE and VVIZU respectively) will be selected instead of top-down-prediction.

Punctuation Several constraints expect a comma or quotation marks between two words. If at least one of the words is virtual, the missing punctuation might still show up later on. Thus we will add respective exceptions to these constraints.

Word form There are constraints accessing the word forms or lemmas. Many of these occurrences are exceptions for certain words. We will ignore them for now, as we neither want too detailed predictions (exploding the search space of the parser) nor do we want to allow idiosyncratic constructions before the licensing word appears in the input. For constraints deemed general enough, exceptions which prevent them to be applied to VNs are added.

Table 1 shows the percentage of prefix structures containing a certain number of VNs, in total and coarsely grouped by part of speech. We also added a line for the number of VNs in the connection path, i.e. only the VNs needed in addition to those already present in the previous prefix, to connect the most recent word to the rest of the structure. This line is roughly comparable to the numbers presented in [4]. Their headless projections in the connection path correspond to our VNs. In both cases connection paths up to length 1 cover 98% of the observed prefixes and 2 cover over 99%. No occurrences of path longer than 4 are observed. Instead of 82.3% of prefixes not requiring any headless projections there, 80.8% of the words can be connected without VNs here.

To estimate whether the minor difference results from differences in language (English VS German) or grammar formalism (phrase structure VS dependency structure), we ran our evaluation on 500 sentences from the WSJ corpus, i.e.

Table 1. Results of experiment 1: The number of VNs needed for prefix structures, only regarding connectedness but no valencies; total VNs: number of VNs per PDA; CP length: only those VNs needed to connect the most recent word; note that the numbers in the columns don't add up: the 4 VNs of a PDA might consist of 3 virtual verbs and 1 virtual nominal

	0	1	2	3	4	5+
total VNs	56.5%	30.1%	10.8%	2.4%	0.2%	0%
verbs	68.8%	26.0%	4.6%	0.7%	<0.1%	0%
nominals	81.2%	18.7%	0.1%	0%	0%	0%
adjectives	97.5%	2.5%	<0.1%	0%	0%	0%
other	99.1%	0.8%	<0.1%	0%	0%	0%
CP length	80.8%	17.0%	2.0%	0.1%	0%	0%
WSJ	79.4%	17.8%	2.7%	0.2%	0%	0%

the same corpus used in [4], converted to dependency structures. The results (cp length only) are shown in the last line of Table 1 and are closer to our results, suggesting that the differences result from the grammar formalism.

6.2 Valency Respecting Prediction

In the previous experiment, a large class of constraint violations were related to unsatisfied valencies. We therefore modified the prefix generation algorithm to keep words in the structure for the following cases:

- Subjects of all finite verbs, virtual or not
- Kernel nouns of prepositions
- Dependents of a conjunction
- Objects of non-virtual words
- The second part of a truncation like 'an- und verkaufen' if the first is observed
- The last part of a conjunction chain, if the first two are observed, e.g. in the sequence 'A, B, C und D', 'und D' would be kept and C skipped, if 'A,B' was observed
- Certain adverbs that fill the role of conjunction in subclauses are kept, if the verbal head of the subclause is already kept for other reasons.

Since the valencies of virtual words are not available to the parser, they all have to be considered optional. Therefore, no objects for virtual verbs are kept.

6.3 Prediction with Folded Structures

The results for PDAs with folding are shown in Table 3. Compared to Table 1, the most noticeable difference is that the percentage of prefixes with two virtual verbs went down by around 3.5%, while the number for 1 virtual verb increased. As expected, there is no change in the zero column number, as folding will not remove the last virtual verb remaining.

Table 2. Results of experiment 2: The number of VNs needed for prefix structures, regarding connectedness and valencies

	0	1	2	3	4	5+
total VNs	37.4%	34.6%	20.7%	5.8%	1.3%	0.2%
verbs	62.6%	30.8%	5.8%	0.8%	<0.1%	0%
nominals	58.5%	37.5%	3.9%	0.1%	0%	0%
adjectives	96.1%	3.8%	0.1%	0%	0%	0%
conjunction	97.9%	2.1%	0%	0%	0%	0%
adverb	99.4%	0.4%	0.2%	0%	0%	0%
preposition	98.3%	1.7%	0%	0%	0%	0%
other	99.2%	0.9%	0%	0%	0%	0%

Table 3. Results of experiment 3: The number of VNs needed for prefix structures, regarding connectedness while folding recursive verb chains

	0	1	2	3	4	5+
total VNs	56.5%	33.3%	9.3%	0.9%	<0.1%	0%
verbs	68.8%	30.2%	1.0%	0.1%	0%	0%
nominals	81.2%	18.8%	0.1%	0%	0%	0%
adjectives	97.5%	2.5%	<0.1%	0%	0%	0%
other	99.1%	0.8%	<0.1%	0%	0%	0%
CP length	81.15%	17.2%	1.6%	<0.1%	0%	0%

6.4 Prediction with Valencies and Folding

The results for the combined algorithm, folding plus valencies, can be seen in Table 4. As expected, the numbers are generally above those of Table 3 and below those of Table 2. Three VNs are sufficient to achieve a coverage of 99%.

One line in the table that might surprise is that prepositions as VNs were observed (1.7%), since a preposition is the leftmost word in a PP. There are three different ways prepositions are predicted: to connect an adverb (arguably a very ambiguous and hard to predict case), to fill a verb's valency for a prepositional object, and to complete a coordination of prepositional phrases, when the first PP and a conjunction like 'und' was already observed.

The question remains, what part-of-speech category combinations among the VNs achieve which coverage. Some possible combinations are shown in Table 5. A set consisting of two nominals and one verb, as used in [6], covers nearly 90%. Adding a VN for adjectives, conjunctions, prepositions and adverbs improve the coverage towards 99%. Whether the parser can actually benefit from this improved coverage, especially of the latter two categories, will have to be evaluated, but is beyond the scope of this paper.

Table 6 shows how VNs are distributed over the different prediction constellations. If a VN serves to connect two different words, e.g. it is the head of a determiner the object of a verb, the leftmost word is selected. The most common reason for prediction is predicting the kernel noun of a PP (18%), followed by predicting the subject of a finite verb (11%), predicting a clause initial conjunction (10%) and connecting a determiner (6%).

Table 4. Results of experiment 4, the complete algorithm: The number of VNs needed for PDAs, regarding connectedness and valencies while folding nested structures

	0	1	2	3	4	5+
total VNs	37.4%	36.7%	20.6%	4.3%	0.9%	0.1%
verbs	62.6%	35.0%	2.3%	0.1%	0%	0%
nominals	58.5%	37.5%	3.9%	0.1%	0%	0%
adjectives	96.1%	3.8%	0.1%	0%	0%	0%
conjunction	97.9%	2.1%	0%	0%	0%	0%
adverb	99.4%	0.4%	0.2%	0%	0%	0%
preposition	98.3%	1.7%	0%	0%	0%	0%
other	99.1%	0.9%	0%	0%	0%	0%

Table 5. Percentage of sentence prefixes that can be represented with a given set of VNs. N: nominal, V: verb, A: adjective, C: conjunction, P: preposition, Av: Adverb

VN-Set	coverage (%)
no VNs	37.4
$2 * N + V$	89.4
$2 * N + V + A$	93.0
$2 * N + 2 * V + A$	94.7
$2 * N + 2 * V + A + C$	96.7
$2 * N + 2 * V + A + C + P$	98.3
$2 * N + 2 * V + A + C + P + Av$	98.7

6.5 Discussion

This study only covers part of the problem of grammar development: We made sure for every prefix of every sentence there is a dependency structure that is scored at least as good as the dependency structure for the final sentence. So far, however, we did not consider the complementary case, namely to make sure that for the best scored partial dependency structure of every sentence prefix there actually is a continuation that would indeed be scored as good. This cannot be examined by a corpus study but only by inspecting actual parser behavior.

7 Conclusions

In this paper we presented a parsing system based on the WCDG framework capable of incrementally parsing into connected dependency structures. The system uses a grammar consisting of weighted constraints as well as a set of virtual nodes. While there is an existing state of the art broad coverage WCD-grammar for German, that grammar is optimized for complete sentences, not for sentence prefixes. We conducted a corpus study to identify what changes to the existing grammar are needed, as well as to identify sets of virtual nodes suitable to cover the observed sentence prefixes. Our experiments complement those conducted by [4] by exploring another grammar formalism and valency driven top-down prediction in addition to connectedness-driven bottom-up prediction.

Table 6. An overview over common reasons for why VNs where included in a PDA; bottom-up prediction to the left, top-down to the right; determined for the PDAs of experiment 4;

reason	% of VNs
connecting a	
- determiner	6.0
- subject	5.9
- clause initial conjunction	5.6
- adverb	4.5
- accusative obj.	3.8
- preposition	3.4

reason	% of VNs
filling the valency:	
- kernel noun of a prep.	18.0
- subject	11.4
- the conjunct under a conjunction	10.3
- accusative obj	5.3
- conjunction to finish an incomplete coord.	3.5
- auxiliary verb	2.7
- predicate	3.4
- subject clause	2.7
- object clause	1.1

References

1. Nivre, J., Hall, J., Nilsson, J., Chanev, A., Eryigit, G., Kübler, S., Marinov, S., Marsi, E.: Maltparser: A language-independent system for data-driven dependency parsing. Natural Language Engineering **13** (2007) 95–135
2. Sturt, P., Lombardo, V.: Processing coordinated structures: Incrementality and connectedness. Cognitive Science **29** (2005) 291–305
3. Bornkessel, I.: The Argument Dependency Model: A neurocognitive approach to incremental interpretation. Max Planck Institute of Cognitive Neuroscience (2002)
4. Lombardo, V., Sturt, P.: Incrementality and Lexicalism: a Treebank Study. In: Lexical Representations in Sentence Processing. John Benjamins (2002) 137–154
5. Demberg, V.: A Broad-Coverage Model of Prediction in Human Sentence Processing. PhD thesis, The University of Edinburgh (2010)
6. Beuck, N., Köhn, A., Menzel, W.: Incremental parsing and the evaluation of partial dependency analyses. In: DepLing 2011, Proceedings of the 1st International Conference on Dependency Linguistics. (2011)
7. Foth, K.A.: Hybrid Methods of Natural Language Analysis. PhD thesis, Uni Hamburg (2006)
8. Brants, T., Hendriks, R., Kramp, S., Krenn, B., Preis, C., Skut, W., Uszkoreit, H.: Das negra-annotationsschema. Negra project report, Universität des Saarlandes, Computerlinguistik, Saarbrücken, Germany (1997)
9. Daum, M., Foth, K., Menzel, W.: Automatic transformation of phrase treebanks to dependency trees. In: 4th Int. Conf. on Language Resources and Evaluation, LREC-2004. (2004)