Incremental Parsing and the Evaluation of Partial Dependency Analyses

Niels Beuck, Arne Köhn and Wolfgang Menzel
Fachbereich Informatik
Universität Hamburg
{beuck, 5koehn, menzel}@informatik.uni-hamburg.de

Abstract

In this paper we discuss options for producing structural descriptions for an input sentence which is not yet completely available. Two existing dependency parsers have been modified to generate sequences of output hypotheses in an incremental manner. The parsing results can be characterized with respect to different criteria like the amount of predicted information, its quality, monotonicity, delay, inclusiveness and connectedness. We propose an evaluation scheme able to capture these properties and apply it to the parsers in different configurations.

1 Motivation

Incremental language processing does not consume its input at once but in a word-by-word manner. A sequence of incomplete, but successively more complete interpretations is generated for an utterance. Such a processing mode is particularly interesting in scenarios where language input evolves over time, like in human-computer or human-robot interaction. Since the input can be processed while it is still incomplete, production time is available as processing time. Moreover it also becomes possible to immediately respond to partial input, either by providing non-verbal feedback to the speaker, taking a turn, or starting an action while a command is still being spoken. Such a behavior requires a system which is able to produce an analysis for partial input. These intermediate results are provided for internal and external use. Internally they can guide the processing of the next input increment. External use includes feedback to previous processing modules and incremental input to subsequent processing modules.

To our knowledge no dependency parser is available so far which is able to generate fully connected intermediate results. Existing incremental dependency parsers wait at least until both words to be connected are available. This renders intermediate structures unconnected in most cases. The integration of new words is often delayed further due to lookahead.

The initial question to be answered in this paper is, therefore, how partial dependency analyses should look like and what information they should contain. This question is addressed in Section 2. In Section 3, possible metrics for the evaluation of partial dependency analyses are discussed. Section 4 shows how partial dependency analyses can be produced with a constraint dependency parser or a shift-reduce parser. In Section 5 the parser output is evaluated, before conclusions are drawn in Section 6.

2 Definitions

A dependency analysis is an directed acyclic graph where the words correspond to nodes and dependencies to edges. Exactly one head, also called regent, and one dependency type, also called label, is assigned to every word in a sentence. Candidates for a regent are the other words from the sentence or a special root node.

The dependency analysis for an incomplete sentence prefix will be called **partial dependency analysis** (PDA) throughout this paper. If only a prefix of the sentence is known, assigning a dependency structure to it is not trivial. The first problem is temporary ambiguity, i.e. the decision about the correct assignment for a word might depend on how the sentence continues. In such cases we cannot determine the correct analysis before the continuation of the sentence becomes available.

This uncertainty aggravates the general problem of global ambiguity, which is omnipresent even in complete sentences.

A second problem is introduced, since the words already known are usually not sufficient to represent the correct analysis. This becomes obvious if the structure of a complete sentence is cut off at an arbitrary position. Then we can distinguish four kinds of dependencies: those with both nodes in the already known prefix, those with an unknown dependent, those with an unknown regent and those lying completely outside the prefix. The most problematic class here is the one with unknown regent, as the dependent is part of the prefix but cannot be assigned correctly to one of the possible regents as defined above, i.e. a known word from the prefix or the root node. There are two possibilities to deal with this problem: delaying the assignment, i.e. not including the respective word into the PDA, or predicting hypothetical nodes for the not yet seen input.

A PDA that assigns a regent to every word in the prefix will be called **inclusive**. An analysis that contains nodes in addition to the ones corresponding to words in the prefix or the root node will be called **predictive**. Correct PDAs have to be predictive to be also inclusive: if the correct regent is not available in the prefix, either a placeholder for it has to be provided, e.g. by prediction, or no regent can be assigned. The minimal extension that is necessary to guarantee inclusion will be called **minimal prediction**. It consists of a single node added to the list of permissible regents. This extra node is predictive in the sense that it does not correspond to a known word from the prefix and it is maximally unspecified, i.e., its surface word form, lemma, part-of-speech and its position beyond the fact that is to the right of the other words are unknown. Even its identity is unspecified, i.e., it could stand for an arbitrary number of words and two dependencies meeting at the predicted node do not necessarily meet in the complete dependency analysis.

Also the new node can only serve as regent, but not as dependent of any other node. This kind of predictive node was previously proposed by Daum (2004) and is called *nonspec* due to its unspecified nature. Assigning nonspec as regent is more informative compared to not including the respective dependent at all: Firstly, a dependency label can be assigned to the attachment and secondly, it can be taken for granted that nonspec is neither one of the known words nor the root node. While delaying the attachment reflects the uncertainty about the correct regent, attachment to nonspec expresses the certainty that the word will not be attached to one of the already known words.

Although minimal prediction facilitates inclusion, it is not sufficient to guarantee the connectedness of a dependency structure, A (partial) dependency analysis is called **connected**, if there is a path of dependencies, ignoring direction, between every two words of the sentence (prefix).

Since nonspec itself does not have a regent, the words assigned to nonspec are not connected to the other nodes of the dependency graph. Such unconnected words cannot be easily related to the rest of the prefix in a semantic interpretation. We will therefore further extend the number of regents to allow **structural prediction**. Now predictive nodes themselves can be assigned to a regent. These so called **virtual nodes** differ from nonspec in that there can be more than one of them, that they require a regent themselves and that each virtual node represents exactly one word from the unknown suffix of the sentence. Features of virtual nodes like their part-of-speech or their order can be specified. Edges between virtual nodes are also possible.

3 Quality Metrics

In the previous section we presented two kinds of prediction which can be used to augment partial dependency analyses. To be able to compare them, we need to quantify and measure their quality.

3.1 Attachment Score

For dependency analyses of complete sentences, quality is usually measured by the attachment score (AS). It is defined as the ratio of words in the sentence that have been assigned to the same regent as in a gold standard annotation of the same sentence (UAS) and, optionally, with the same label (LAS).

There are several difficulties in applying these measures to partial dependency analyses, especially to those including predictions: First of all, there are no gold standard annotations available for sentence prefixes, only for complete sentences. This problem can be addressed in two different ways: either by annotating a corpus of sentence prefixes, or by generating prefix annotations from

existing full sentence annotations. As there are multiple prefixes for every sentence, annotating prefixes requires considerable effort compared to annotating complete sentences. In addition, temporary ambiguity might result in multiple plausible annotations for the same prefix. The same problem occurs if prefix annotations are extracted from a corpus of complete annotations. Two sentences might share a prefix but assign a different syntactic structure to the words in the prefix.[1] The resulting corpus would then contain two "correct" analyses for the same sequence of words. Additionally, the interpretation for the complete sentence might not be the most plausible interpretation for each of its prefixes. However, a corpus large enough might level out these implausible prefix annotations by a greater number of plausible annotations for similar syntactic constructions. A parser that always chooses the more common structure for a prefix would then obtain a better score. In this paper we will use existing dependency annotations.

Complete dependency analyses can be rated with only one score instead of two values like precision and recall because the number of nodes is fixed for a given sentence in contrast to the number of nodes in a phrase structure graph. For PDAs, however, the number of words in the structures to be compared might vary, depending on how many of them have been included or predicted. Therefore, the accuracy measure for PDAs has either to be split into a precision and a recall part, or only inclusive PDAs with no prediction beyond minimal prediction can be considered.

Structurally predictive PDAs can be reduced to minimal predictive ones by interpreting all attachments to predicted regents as minimal predictive attachments and ignoring all attachments of predicted dependents. For cases of non-inclusiveness, where words only remain unattached as long their heads are not yet available, the missing attachments can be interpreted as nonspec attachments[2]. This way we can guarantee a fixed number of dependents for a prefix.

Let A be an annotation consisting of dependency arcs $(dependent, label, regent)$. Then the annotation A_P for a prefix P is:

$A_P = \{(d,l,r) \in A | \ d \in P \wedge r \in P \cup \{\mathbf{root}\}\} \cup \{(d,l,\mathbf{nonspec}) | \ d \in P \wedge \exists_x((d,l,x) \in A \wedge \neg x \in P \cup \{\mathbf{root}\})\}$

A general PDA B for a prefix P can be reduced to a minimal predictive and inclusive PDA:

$\hat{B} = \{(d,l,r) \in B | \ d \in P \wedge r \in P \cup \{\mathbf{root}\}\} \cup \{(d,\mathbf{nolabel},\mathbf{nonspec}) | \ d \in P \wedge \neg\exists x.(d,\_,x) \in B\} \cup \{(d,l,\mathbf{nonspec}) | \ d \in P \wedge \exists v.(d,l,v) \in B \wedge virt(v)\}$

With this normalization we can assign an attachment score to a PDA as usual. This measurement does not reward prediction beyond the minimal prediction, but provides a common ground for all inclusive partial dependency analyses.

For the Prefix "John" of the sentence "John buys a book" the annotation $\{A = (John, SUBJ, buys), (buys, S, root)...\}$ and the predictive analysis $B_1 = \{(John, SUBJ, [virt]), ([virt], S, root), ...\}$ would both be normalized to $\{(John, SUBJ, nonspec)\}$, resulting in an AS of 100%. The non-inclusive and empty analysis $B_2 = \{\}$ would be normalized to $\{(John, nolabel, nonspec)\}$, resulting in a UAS of 100%, but a LAS of 0%.

3.2 Accumulating prefix scores

Incremental parsing does not produce a single prefix analysis, but sequences of them. Simply accumulating the accuracies for all the words in all prefixes would introduce a strong bias in favor of the earlier tokens: a word appearing early in a sentence will have a greater influence on the overall score than a later one, giving it more weight in the accumulated score.

Therefore, we apply a sliding window to the sequences of PDAs. For every word the attachment status is determined not only for the prefix it first appears in (and the final result), but also for a fixed number of prefixes in the vicinity of the first appearance. This allows us to investigate the temporal evolution of a word's attachment as well as to give all words the same weight.[3]

[1] Among 15000 sentences from the Negra corpus more than 5000 share a prefix with another one, which is annotated differently. 81% of these shared prefixes are of length one, 15% of length two and 4% longer than two.

[2] However, no dependency label can be assigned in such a case. Also, incremental parsers can use lookahead to incorporate features of later words into the decision for a word. This interferes with the re-interpretation of missing attachments as attachments to nonspec, since it would be wrongly assumed that all the words in the lookahead window should be attached to nonspec. Therefore, such results are better dealt with by specifying the fixed lookahead size or providing a separate recall value.

[3] modulo effects at the start and end of the sentence, depending on the window size

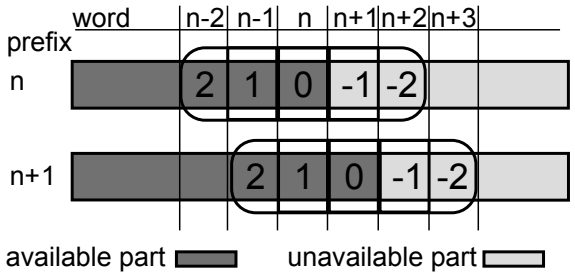


Figure 1: Sliding window for prefix n and $n+1$ with relative word indices

Note, however, that for incremental systems using lookahead there might not be a single PDA for every input increment. As the lookahead window must first be filled up before any output can be generated, no explicit output will be produced for the first n input increments. As the final input increment fills the lookahead window of the last $n+1$ words, there are $n+1$ output increments for it. This can be compensated by adding n empty output increments at the beginning and by keeping only the final analysis for the last input increment.

We can determine three different attributes for an attachment: its correctness (including the label or not), its status (i.e., whether it is included, not included, a minimal or a structural prediction) and its stability (i.e., whether it differs from a later PDA). There might be a difference in the degree to which the predictive attachments are specified. Here, we distinguish between minimal prediction without dependency label, with dependency label and structural prediction.[4]

For every PDA the window is centered at the rightmost input word, so that the previous words are assigned to slots with ascending numbers, as illustrated in Figure 1. The correctness C_{minP} for a given slot for a given prefix is determined as discussed above for the attachment score where nonspec matches any non-included word. Precision P_{minP} and recall R_{minP} can then be calculated by dividing by the number of included words in the PDAs or the total number of encountered words respectively.[5] Both can be averaged for the window (cf. Figure 2).

With respect to structural prediction two additional aspects have to be considered. First of all, the predicted words have to be mapped to words from the gold standard in a one-to-one correspondence. While different mappings might be possible, the mapping bm with the best overall accuracy is chosen. This optimum might depend on whether accuracy is measured labeled or unlabeled, we use unlabeled attachment. Given that a virtual node can partake in more than one dependency edge, i.e. it has one regent and an arbitrary number of dependents, a virtual node can possibly be mapped even if some of these attachments are incorrect. Mappings that share no edge with the gold standard annotation are not considered. Therefore, some virtual nodes might remain unmapped. As the optimal mapping might change as the prefix of a sentence grows, the same virtual node can be mapped to different words for consecutive PDAs. Per definition, all predictions still left in the final result are incorrect.

Secondly, with structural predictions words can be assigned to a regent before they become part of the available prefix. This corresponds to a negative slot index in the window. The sliding window, however, is applied to the positions of the words in the gold standard annotation, not in the PDA, as virtual nodes are not located at a specific position. Thus, they only have the potential to be captured by the sliding window if being successfully mapped. If mapping fails, the virtual node has no well-defined position. Therefore, only the recall of structural R_{strP} prediction, but not the precision P_{strP}, can be determined by means of a sliding window alone. For words with a negative slot number, we calculate the correctness $C_{strP,V}$ independently of slots for all predicted words and combine it with the correctness of the non-negative slots to obtain an accumulated precision, as defined in Figure 2

3.3 Stability

The stability score for a given slot is calculated like precision, but the PDA is compared to the final annotation found by the parser instead of the gold standard annotation.

3.4 Connectedness

Early semantic interpretation requires to integrate incoming words into a connected structure immediately. We quantify the degree of connectedness of a PDA by means of its average **fragmentation**,

[4]Even a more comprehensive prediction including the lexical features of a word or its surface form would be possible. This has not been considered here as the choice of the correct lexical reading is also not covered by the usual definition of the attachment score.

[5]This number depends on the slot number: all words are encountered by slot 0, but slot n will not encounter the last n words of a sentence.

$$R_{minP} := \frac{\sum_{p \in P_s} \sum_{i \in W} C_{minP}(i, pda_p, |p|)}{\sum_{i \in W}(|s| - |i|)}$$

$$P_{minP} := \frac{\sum_{p \in P_s} \sum_{i \in W} C_{minP}(i, pda_p, |p|)}{\sum_{p \in P_s} |pda_p \cap W|}$$

$$R_{strP} := \frac{\sum_{p \in P_s} \sum_{i \in W} C_{strP}(i, pda_p, |p|, bm(pda_p))}{\sum_{i \in W}(|s| - |i|)}$$

$$P_{strP} := \frac{\sum_{p \in P_s}(\sum_{i \in W_+} C_{strP}(i, pda_p, |p|, bm(pda_p)) + \sum_{v \in V_{pda_p}} C_{strP,V}(v, pda_p, bm(pda_p)))}{\sum_{p \in P_s}(|pda_p \cap W_+| + |V_{pda_p}|)}$$

$$C_{minP}(sID, pda, n) := \begin{cases} 1 & \text{if } reg_{pda}(n - sID) = reg_{gold}(n - sID) \\ 1 & \text{if } reg_{pda}(n - sID) = nonspec \wedge reg_{gold}(n - sID) > n \\ 0 & \text{else} \end{cases}$$

$$C_{strP}(sID, pda, n, map) := \begin{cases} 1 & \text{if } map(reg_{pda}(n - sID)) = reg_{gold}(map(n - sID)) \\ 0 & \text{else} \end{cases}$$

$$C_{strP,V}(vn, pda, map) := \begin{cases} 1 & \text{if } map(reg_{pda}(vn)) = reg_{gold}(map(vn)) \\ 0 & \text{else} \end{cases}$$

Figure 2: Definitions for precision and recall for a single sentence, where P_s is a the set of all prefixes of a sentence s, W the slot-ids of the used window, W_+ the non-negative ones, pda_p the analysis for a prefix p, V_{pda} the virtual words used in pda and $|pda \cap W|$ the amount of arcs in an analysis covered by the window, i.e. the number of included words. $bm(pda)$ is the best mapping as defined in Section 3.2.

defined as the average number of tree fragments in addition to the first one.

This is an indication of how many attachments have to be changed at least, to produce a connected tree. As minimal predictive attachments do not predict whether they attach to the same word, each such attachment has to be counted as a potential root of an additional tree fragment. Punctuation marks are never integrated into the dependency graph, and therefore not be considered. The average fragmentation number can then be compared to the fragmentation of the gold standard annotation.

4 Implementation

In this section we will present two approaches for incremental parsing which produce partial dependency analyses that are both inclusive and predictive. We modified two existing parsing systems WCDG[6] and MaltParser[7] to generate incremental output.

4.1 WCDG

Weighted Constraint Dependency Grammar (WCDG) is a framework, which maps dependency parsing to the problem of constraint optimization (Schröder, 2002). Menzel (2009) proposed an incremental parsing strategy for WCDG based on the repair based algorithm frobbing (Foth, 2006). It tries to improve an initial structure through a sequence of conflict driven transformation steps. To perform incremental parsing this algorithm can be applied to the prefix of a sentence, and the generated structure (plus an arbitrary attachment for the new words) is used as a starting point for the analysis of the extended prefix. This approach is non-monotonic, as the previous PDA provides only a starting point for the next search step, but the resulting PDA does not need to include all the arcs of its predecessor. WCDG is able to profit from information contributed by external modules (Foth, 2006). We used only use the most essential ones: a PoS tagger and a PP attacher, for our experiments.

Nonspec

As frobbing produces inclusive dependency analyses where a regent is assigned to every word, the system has to assign a regent even in the absence of the intended one (c.f. Section 2). A suitable attachment point has to be made available. A minimal prediction with a nonspec node serves exactly this purpose.

[6] https://nats-www.informatik.uni-hamburg.de/view/CDG/

[7] http://maltparser.org

With nonspec, the grammar has to be changed in two regards. First, a constraint is added to slightly penalize nonspec attachments. It guarantees that a nonspec attachment is only chosen if no suitable real regent is available.

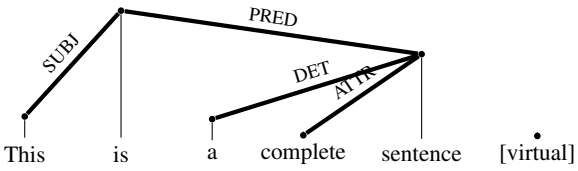
A second change adds guards to the constraints to prevent non-existing attributes of the regent from being accessed. These guards are specified in a way that they replace the non-specified feature with an optimistic estimation. For example a query for a comma between the two ends of a dependency edge would return *true* for constraints demanding a comma, while the same query would return *false* in a constraint that forbids a comma (unless there is already a comma in the known part of the prefix).

Virtual Nodes

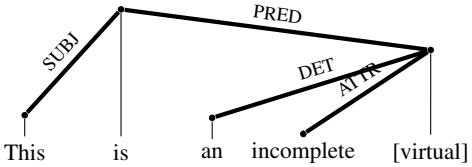
While this implementation of incremental dependency parsing accomplishes minimal prediction, it does not exhaust the potential for syntactic prediction of a given grammar. Constraints demanding the existence of certain words or their lexical features are either prevented from accessing those features, or alternatively their violation, like an unsatisfied verb valency, is simply accepted because no less penalized alternative is available. In particular, no proof is required that and how a predicted regent itself could be integrated into the rest of the dependency structure without violating additional constraints. To extend the range of prediction in dependency analyses, the concept of virtual nodes as defined in Section 2 is used.

Since the frobbing search algorithm is not able to add or remove words to or from the constraint problem, a maximal set of potentially useful predictive nodes has to be introduced prior to search. As with nonspec, attachments to and from virtual nodes are penalized slightly. Virtual nodes which are not integrated into an analysis stay unconnected and are considered unused. As a result, used virtual nodes are per definition always attached to another node. Unused virtual nodes are assigned to the root node with the empty label as dependency type. They may not be assigned as regents to other words. This is enforced by a hard constraint in the grammar. They are not considered part of the sentence and can safely be removed from an analysis without altering its meaning.

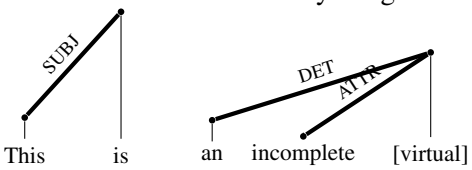
Example for an unused virtual node:



Example for a used virtual node:



Example for a partially used virtual node, a constellation that is not allowed by the grammar:



Virtual nodes, once added to the constraint problem technically behave like other words. Their predictive nature is not visible to the search algorithm, as the topology of the search space remains the same. All restrictions mentioned above are enforced via constraints in the grammar. To be able to distinguish between virtual and non-virtual nodes in a constraint, a new attribute *virtual* is defined. A corresponding predicate can be invoked by a constraint definition. With this approach, prediction, i.e. the inclusion of virtual nodes into the dependency structure, is purely constraint driven.

We can distinguish between two kinds of prediction, bottom-up and top-down. In bottom-up prediction, the inclusion of a predictive node is driven by an unconnected word, for which every other integration would result in constraint violations. Top-down prediction is conflict driven in that a specific constraint violation indicates the need for an additional dependent of an existing word, as it is the case for verb valencies. By providing the search algorithm with a set of predictive nodes for potential use, predictive partial results can be generated without further adaption of the algorithm. All that is needed is adding candidates for dependency arcs for the virtual nodes to the search space and extend the grammar, as discussed above.

Replacement of virtual nodes

After a prediction has been included into an incremental parsing step, it has to be replaced later on with a word from the input, once a fitting word be-

comes available. This can in principle be achieved by the search algorithm, but many transformation steps might be needed to properly integrate it into the existing structure. An alternative consists in checking for each new word prior to search, whether it can fill one of the used virtual nodes, instead of adding it as a separate word to the structure.

To determine whether a replacement is successful, the ratio of penalties assigned by the grammar before and after replacement is compared to a threshold. If at least one successful replacement has been found, the one with the highest score is used as starting point for the next search step. Otherwise the word is appended as usual.

4.2 MaltParser

MaltParser (Nivre et al., 2007) is a dependency parser which provides an incremental algorithm but no incremental output in the sense of intermediate analyses for prefixes. We, therefore, had to modify the parser in a way that allows us to extract partial dependency analyses from its hypothesis space. For that purpose the set of already submitted dependency arcs is recorded immediately before the next word in the input buffer is read, while yet unattached words are considered to be attached to nonspec. This allows us to recover the PDAs for every increment.

There are several algorithms available for MaltParser. The best choice depends on coverage of non-projectivity, eager arc attachment and explicit root handling. As the evaluation is done for German, a language with a comparably high degree of non-projective constructions, it is mandatory to use a version which is able to deal with non-projectivity.

In general, the shift-reduce approach used by MaltParser does not guarantee an arc to be built as soon as both nodes are available. As the attachment reduces the token from the stack rendering it unavailable to further attachments, dependents to the right of their head cannot be attached before all their dependents have been included into the structure. Nivre (2003) proposed a so called arc-eager approach, which splits the *right-reduce* action into a *right-arc* and a *reduce* action. This modification allows an immediate attachment, once head and dependent are available.

There are two ways to deal with root attachment, either as an explicit attachment via an arc building action or by waiting until the sentence has been completely parsed and attaching all words still left unattached to the root. For our purpose we need the explicit root attachment approach to be able to distinguish temporarily unattached words (interpreted as nonspec attachment) from root attachments. The explicit root attachment implemented by MaltParser has proven not to be exhaustive. Especially punctuation tokens are always left unattached. As those are always attached to the root, it is easy to deal with them separately. For other words that are left unattached despite explicit root handling there is no way to detect whether they will stay unattached until the end of the sentence. The impact of this problem on accuracy, however, is minimal, as for these words the root attachment would often be incorrect as well.

From the algorithms fulfilling these requirements we choose the 2-planar algorithm (Gómez-Rodríguez and Nivre, 2010), as it provides the best performance for German and does not require post-processing to recover non-projective links. It uses an approach with two stacks and an additional parsing action to *switch* between them. Although this does not allow it to parse general non-projective structures, all 2-planar non-projective structures can be dealt with, which covers most non-projectivities in most natural languages, e.g., more than 98% for German. For more details see Gómez-Rodríguez and Nivre (2010).

4.3 Differences between WCDG and MaltParser

The biggest difference is that MaltParser is trained on a tree-bank while WCDG uses a manually generated dependency grammar together with trained external components.

Both parsers apply an incremental algorithm in the sense that information from a previous analysis are used to calculate the analysis for the extended prefix, but apply different strategies to deal with temporary ambiguity as defined in Beuck et al. (2011). While WCDG applies reanalysis, resulting in timely but non-monotonic output, MaltParser applies lookahead, resulting in monotonic but delayed output.

5 Evaluation

5.1 Setup and data

In this section we will compare different configurations of MaltParser and WCDG by evaluating

them with the metrics proposed in Section 3. Evaluation has been carried out on 500 German sentences from the Negra corpus converted to a dependency structure. MaltParser was trained on 15000 different sentences from the same corpus.

Also, the parsers are compatible with different strategies of incremental PoS tagging. While MaltParser is restricted to taggers with best guess or lookahead strategies, WCDG is able to integrate multi-tagging and non-monotonic tagger output. Based on the evaluation of incremental PoS taggers in Beuck et al. (2011), we chose the TnT tagger[8] with multi-tagging and retagging of each prefix for WCDG, while MaltParser is combined with SVMTool[9] using a lookahead of one or zero, depending on the configuration.[10] As MaltParser accuracy is often reported on gold tagged input, we also provide these numbers for a comparison with already published non-incremental results.

We evaluated MaltParser configurations with a total lookahead between zero and four, as well as incremental WCDG configurations with nonspec (NS), virtual nodes (VN), and both mechanisms activated (VN+NS). All these configurations are evaluated with the scheme for minimal prediction (P_{minP} and R_{minP}). In addition, the WCDG-VN configuration is evaluated in terms of structural prediction (P_{strP} and R_{strP}).

5.2 Discussion

Table 1 contains final accuracy, as well as precision and recall values integrated over a window of size 9. Figure 3 shows the temporal evolution of accuracy and stability within the window for different parser configurations. In these figures the different behavior of MaltParser and WCDG become apparent.

Due to the monotonic nature of MaltParser, the only possible change in subsequent output increments is the replacement of minimally predictive attachments by fully specified ones. Thus, the average accuracy of attachment decreases after the initial appearance of a word. In contrast to this, the accuracy can even rise over time if reanalysis is allowed as in WCDG. Here, the stability of initial attachments is only 70%, i.e., 30% of the attachments have been changed later on.[11]

A noteworthy observation is that WCDG proposes significantly more erroneous initial attachments to available words, where a predictive attachment would be a better choice. Obviously, it is too eager to attach words to available regents, but is able to recover in many cases by means of reanalysis.

The delayed output of the configurations with lookahead leads to a smaller number of slots[12] in the window having received any attachments. This is reflected in a reduced recall.

Structural prediction with WCDG leads to an increased recall score, 48.6% compared to the best recall without structural prediction of 46.2% for MaltParser) and 44.3% for WCDG. The precision, however, is reduced, but it should also be noted that a structural predictive attachment contains more information. If we ignore this additional information and interpret the virtual nodes only in a minimal predictive sense, precision and recall are higher than in the configuration with nonspec. The real benefit of structural prediction can be seen in the significantly reduced fragmentation, as indeed the PDAs are connected to a similar degree as the gold standard annotations for the full sentences (which has a fragmentation of 0.17%).

6 Related Work

To our knowledge, partial dependency analyses have not been investigated previously in detail. Work on incremental dependency parsing like Nivre (2004) was focused on the incrementality of the algorithm, not on providing an incremental interface. Therefore, the output of intermediate results was not a primary goal. In other cases, like Menzel (2009), the evolution of partial analyses has been studied, but no broad scale evaluation has been carried out.

Regarding connected partial analyses in other grammar formalisms, Demberg and Keller (2008) presented a variant of the tree adjoining grammar (TAG)formalism that is able to incrementally produce fully connected prefix analyses. In this approach prediction plays a strong role, too. Top-

[8] http://www.coli.uni-saarland.de/ thorsten/tnt/

[9] http://www.lsi.upc.edu/ nlp/SVMTool/

[10] Another reason for not using SVMT for WCDG, besides performance in different incremental tagging modes, is that SVMT is not able to provide tag percentages, which are needed as constraint weights in WCDG.

[11] In fact there is also a kind of non-monotonicity in MaltParser, if we interpret the unattached words as "to be attached to a not yet available word". These are reinterpreted as being attached to root in the final result, leading to stability reduction of 5%

[12] The few assignments in slots 0-3 in Figure 3d are due to an end-of-sentence effect, where the lookahead window is filled preliminarily.

parser	configuration	final AS unlabeled	final AS labeled	Average Fragment.	Sliding Window Precision	Sliding Window Recall	Precision of struct. pred.
Minimal Prediction:							
WCDG	NS	**87.02%**	**84.95%**	1.00	78.22%	43.28%	-
	VNs	86.74%	84.57%	1.00	79.98%	44.25%	-
	VNs + NS	86.58%	84.43%	1.00	79.95%	44.23%	-
Malt	LA 0+0	82.28%	78.99%	1.45	83.57%	46.24%	-
	LA 1+0	84.27%	80.65%	1.36	85.27%	38.59%	-
	LA 2+0	84.62%	80.98%	1.26	86.00%	30.75%	-
	LA 2+1	85.04%	81.74%	1.10	86.81%	23.28%	-
	LA 3+1	85.06%	81.66%	0.98	**87.11%**	16.06%	-
Structural Prediction:							
WCDG	VNs	86.74%	84.57%	**0.16**	77.05%	**48.55%**	63.46%
Gold Tagged:							
Malt	gold tags	88.76%	86.25%	-	-	-	-

Table 1: Evaluation of incremental WCDG with different configurations regarding prediction; Lookahead numbers are given as "parser LA + tagger LA"

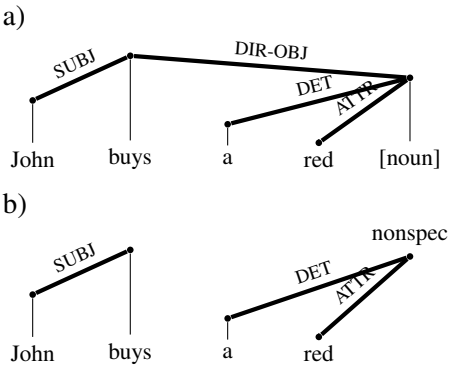


Figure 4: A connected (a) and an unconnected (b) PDA

down prediction is facilitated through substitution nodes in lexical entries, e.g. verbal valencies. Bottom up prediction is achieved by means of connection paths, i.e. the need for additional nodes to connect a subtree to the rest of the structure. A comparison with our results by applying the proposed metrics on derivation trees of TAGS is beyond the scope of this paper but a promising topic for further research.

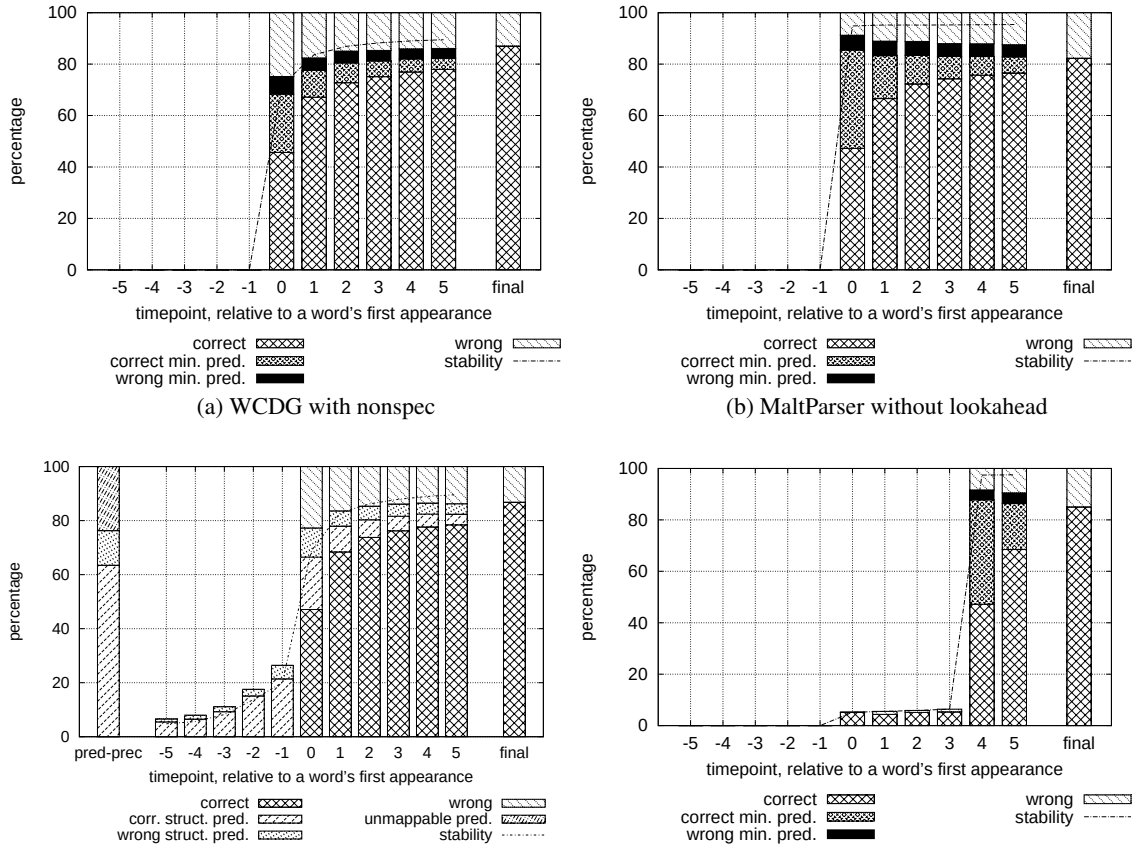
The metrics in this paper only capture syntactic similarity, but not the utility of an analysis for an application task. Eventually, a more semantically oriented measure would be desirable, which reflect the amount of semantic information conveyed by a structure. The sentence prefix "John buys a red", for example, contains the information $buys(John, X)$ and $color(X, red)$. Since such an information can be more easily extracted from a predictive dependency analysis like the one in Figure 4 a) compared to the not connected analysis (4 b)), it would be desirable to assign a higher recall value to a). An application oriented measure for prefix analyses is defined by Schlangen et al. (2009) where several variants for incremental reference resolution are discussed. They are, however, only applicable for utterances with a single reference.

7 Conclusions

In this work we presented a definition of partial dependency analyses that allows us to derive fully connected structures by introducing predicted nodes into the dependency graph. It was discussed how the attachment score metric can be extended to also cover such prefix analyses. In addition, a windowing approach was adopted to analyse the temporal evolution of incremental output sequences in more detail.

Using these measures, two existing dependency parsers have been compared. Obviously, there is still a large number of parameters left unexplored, especially for the instantiation of virtual nodes. Eventually, it will be interesting to study possible similarities between psycholinguistic findings about garden path or reanalysis phenomena and the behaviour of the presented architectures.

This work was funded by the Deutsche Forschungsgemeinschaft (DFG) as part of the International Graduate Research Group CINACS.



(a) WCDG with nonspec

(b) MaltParser without lookahead

(c) WCDG with virtual nodes

(d) MaltParser with a total lookahead of 4

Figure 3: Scores for a sliding window with 9 slots; slot 0 holds a words first appearance in the input; the earlier slots to the left are only filled for the configuration with structural prediction; the leftmost bar is the precision of the attachment of virtual nodes (pred-prec), the rightmost one is the AS for the complete sentence; all scores are unlabeled

References

Niels Beuck, Arne Köhn, and Wolfgang Menzel. 2011. Decision Strategies in Incremental PoS Tagging. In *Proceedings of NODALIDA 2011*.

Michael Daum. 2004. Dynamic dependency parsing. In *In Proceedings of the ACL 2004 Workshop on Incremental Parsing*.

Vera Demberg and Frank Keller. 2008. A psycholinguistically motivated version of tag. In *Proceedings of the ninth international workshop on tree adjoining grammars and related formalisms*.

Kilian A. Foth. 2006. *Hybrid Methods of Natural Language Analysis*. Ph.D. thesis, Uni Hamburg.

Carlos Gómez-Rodríguez and Joakim Nivre. 2010. A transition-based parser for 2-planar dependency structures. In *Proceedings of ACL 2010*.

Wolfgang Menzel, 2009. *Recent Advances in Natural Language Processing V*, chapter Towards radically incremental parsing of natural language, pages 41–56. Number 309 in Current Issues in Linguistic Theory. John Benjamin's Publisher.

J. Nivre, J. Hall, J. Nilsson, A. Chanev, G. Eryigit, S. Kübler, S. Marinov, and E. Marsi. 2007. Maltparser: A language-independent system for data-driven dependency parsing. *Natural Language Engineering*, 13:95–135.

Joakim Nivre. 2003. An Efficient Algorithm for Projective Dependency Parsing. In *Proceedings of IWPT 03*.

Joakim Nivre. 2004. Incrementality in deterministic dependency parsing. In *Incremental Parsing: Bringing Engineering and Cognition Together, Workshop at ACL 2004*.

David Schlangen, Timo Baumann, and Michaela Atterer. 2009. Incremental reference resolution: the task, metrics for evaluation, and a bayesian filtering model that is sensitive to disfluencies. In *Proceedings of SIGDIAL 2009*.

Ingo Schröder. 2002. *Natural Language Parsing with Graded Constraints*. Ph.D. thesis, Uni Hamburg.