Predictive Incremental Parsing and its Evaluation

Niels BEUCK [a], Arne KÖHN [a] and Wolfgang MENZEL [a]
[a] *Fachbereich Informatik*
Universität Hamburg
{*beuck, koehn, menzel*}*@informatik.uni-hamburg.de*

Abstract. For incomplete sentences, different structural descriptions with different levels of prediction are possible. 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.

Keywords. Incremental Parsing, Dependency Grammar, Evaluation, jwcdg, MaltParser

1. Introduction

Humans do not perceive sentences at once but in a word-by-word manner. There is evidence that they have available connected syntactic interpretations as well as predictions about the upcoming input at every point in time while reading a sentence [1].

For several reasons, such a processing mode is also beneficial for artificial NLP systems. It 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 of an incremental processing module are provided for external or internal use. External use includes feedback to previous processing modules and incremental input to subsequent processing modules, including feedback to the human interlocutor, internally they can guide the processing of the next input increment. For an NLP system to work incrementally, all of its components have to work incrementally. In this work we focus on dependency parsing, an important module in many NLP systems.

A fundamental problem in incremental dependency parsing is that the output elements of dependency parsing, the dependency edges, are each related to two

input elements, i.e. words in the sentence. Even if edges are produced as soon as both words they connect are available, the output does not grow at the same rate as new input becomes available but can be significantly delayed. Also, intermediate dependency structures are often fragmented as words stay unconnected until their head appears. This happens, e.g., in incomplete noun phrases, where determiners and left adjectival attributes stay unconnected, until the head noun appears. It is, however, not limited to phrases under construction. In German subclauses, for example, the verb comes last and all its arguments would stay unconnected until it finally becomes available.

In principle, intermediate dependency structures can be augmented by predicting missing parts of the sentence. This way, an incremental dependency parser provides a connected dependency graph as output for every sentence prefix and syntactic information arguably already present in a sentence prefix will be expressed explicitly in the intermediate parser output. The quality of such predictive partial dependency analyses is, however, not easily evaluated against existing gold standard annotations. Also, the dynamics of incremental output is not captured by the established evaluation metrics for dependency parsing. We, therefore, propose a new approach to the evaluation of output of an incremental parser, i.e. the evaluation of a sequence of partial dependency analyses.

The goal here is to provide means to compare incremental output sequences of dependency parsers for different levels of prediction and different output dynamics. In Section 2 the question is discussed how partial dependency analyses can look like and what information they should ideally contain. We investigate two incremental dependency parsers, jwcdg [2] and MaltParser [3]. An overview over how they work and what kind of incremental output they provide is given in Section 3. In Section 4, possible metrics and gold standards for the evaluation of partial dependency analyses are discussed. In Section 5 the parser output is evaluated, before conclusions are drawn in Section 7.

This contribution is an extended and modified version of a paper presented at Depling 2011 [4].

2. Partial Dependency Analyses

A **dependency analysis** is a directed acyclic graph where the words correspond to nodes and dependencies to edges. Exactly one head and one dependency type, also called label, is assigned to every word in a sentence. The dependent is said to be attached to the head. The head of a word could be one of the other words in the sentence or a dedicated root node. The set of possible labels depends on the annotation scheme. Typically, dependency graphs are required to be connected, with only a single word attached to the root node. A **partial dependency analysis (PDA)** is a dependency analysis for an incomplete sentence prefix. A (partial) dependency analysis is called **connected** if there is a path of dependencies, ignoring direction, between every two words of the sentence (prefix).

If only a prefix of a sentence is known, often no unique dependency structure can be assigned to it. The problem here is twofold. On the one hand, the decision about the correct assignment for a word might depend on how the sentence will

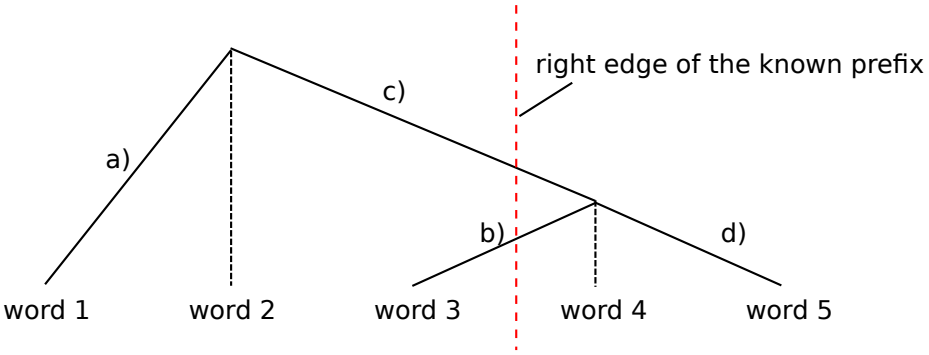


Figure 1. The four different kinds of dependency edges in relation to the prefix boundary

continue, i.e., it is **temporarily ambiguous**. In such cases, we cannot determine the correct analysis before the continuation of the sentence becomes available. Thus, it aggravates the general problem of global ambiguity which is omnipresent even in complete sentences.

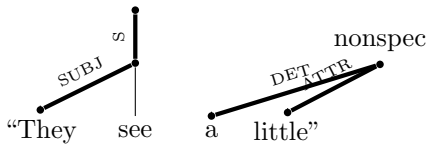
On the other hand, the words already known are usually not sufficient to represent a correct analysis. This becomes obvious if the structure of a complete sentence is cut off at an arbitrary position, as seen in Figure 1. Four kinds of dependencies can be distinguished: those with both nodes in the already known prefix (a), those with an unknown head (b), those with an unknown dependent (c) and those lying completely outside the prefix (d). The most problematic class here is (b), where the dependent is part of the prefix but cannot be attached correctly to one of the available heads as defined above, i.e. a known word from the prefix or the root node. In the example in Figure 1, word 3 is attached to word 4 in in the complete structure, however, that word is not available in the prefix. There are two possibilities to deal with this problem, either by delaying the assignment, i.e. not including the respective word into the PDA, or by predicting hypothetical nodes for the not yet seen input and attaching the word to one of them.

A PDA that assigns a head 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**. For many sentence prefixes, correct PDAs have to be predictive to be also inclusive: if the correct head is not available in the prefix, either a placeholder for it has to be provided, e.g. by prediction, or no head can be assigned.

The minimal extension, called **minimal prediction**, achieving this is to add a single node to the set of permissible heads. The extra node does not correspond to a known word from the prefix, it is maximally unspecified and therefore called **nonspec** [5]. 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 words attached to the predicted node do not necessarily share the same head in the complete dependency analysis. Also, the new node can only serve as head, but not as dependent of any other node.

Assigning nonspec as head 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 head, attachment to nonspec expresses the certainty that the word will not be attached to one of the already known words. A dependency edge attaching nonspec can be considered an underspecified version of an edge in a later PDA where the same dependent word is attached to a newly available word.

Although minimal prediction facilitates inclusion, it is not sufficient to guarantee the connectedness of a dependency structure. Since nonspec itself does not have a head, 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, as seen in the following PDA:



Here, the information that something little is seen cannot be extracted directly from the dependency structure.

Connectedness of a PDA can be achieved by extending the set of available nodes further by adding so called **virtual nodes (VNs)**. They differ from nonspec in that there can be more than one of them and that they require to be attached to a head themselves. Dependency edges between virtual nodes are also allowed.

In addition to establishing a dependency relation, virtual nodes can also be used to carry predictions about the words themselves, like the part-of-speech, lemma or other lexical features. While their precedence relation, i.e. the order among the anticipated words, could theoretically be predicted, here we focus on the prediction of dependency structure and only assign part-of speech information to the VNs. With this approach, called **structural prediction**, partial dependency analyses can be constructed that resemble the dependency analysis of a whole sentence. Theoretically, the PDA for a one word sentence prefix could already contain nodes for every other word in that sentence, anticipating the complete dependency structure. This is, of course, not a realistic output to be made by a parser.

There are two related questions here: what can be predicted and how should gold standard PDAs look like. The answer to the first question is the answer to the second question, as the gold standard PDA should contain exactly that information that could theoretically be predicted. Structure can be predicted if its absence in a PDA would violate some linguistic constraints. This comes in two forms: to connect new word in sentence prefix to the rest of the structure (bottom-up prediction), as seen in Figure 2, or to fulfill obligatory valencies (top-down prediction), as seen in Figure 3.

Connectedness can be defined in two ways, either including the sentence root or neglecting it. In the first case, a full sentence is predicted, in the second just a

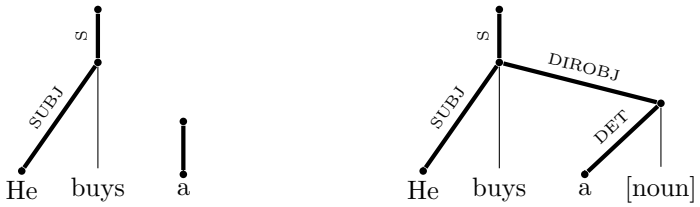


Figure 2. An example for bottom-up prediction: The determiner cannot be attached to any word directly, but can be included if a noun is predicted

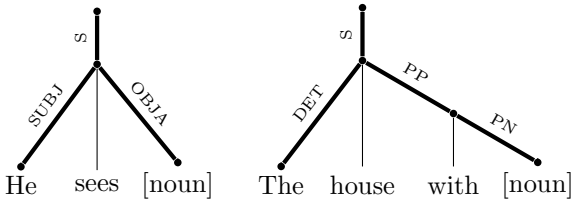


Figure 3. Examples for top-down predictions of an object and the kernel noun of a preposition

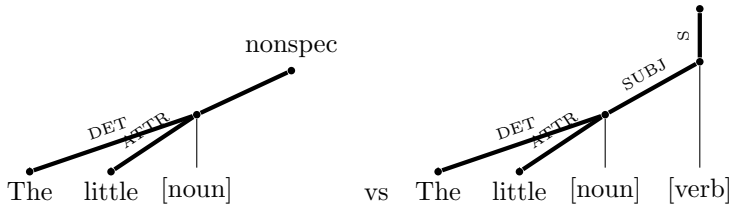


Figure 4. Two levels of completion: Sentence fragment versus fully connected sentence structure

(connected) fragment. This is exemplified given the sentence prefix "The little" as seen in Figure 4. To obtain a connected dependency structure, it is sufficient to predict a noun here. A complete sentence would also have a verb connected to the root node. To express that a word is just the root of a fragment, not the root of the sentence, it is attached to nonspec. Minimal and structural prediction can be mixed this way.

3. Implementations for Predictive Incremental Parsing

In this section two approaches for incremental parsing into a sequence of partial dependency analyses are presented: jwcdg[1] and MaltParser[2].

3.1. jwcdg

jwcdg is a parser that implements the Weighted Constraint Dependency Grammar (WCDG) framework. Dependency parsing is mapped to the problem of constraint

[1] `https://nats-www.informatik.uni-hamburg.de/view/CDG/`
[2] `http://maltparser.org`

optimization [6]. A grammar consisting of defeasible constraints defines a ranking among all possible dependency analyses where the score of a dependency analysis is the product of the penalties of all violated constraints. Parsing in WCDG is the process of finding the best scored analysis.

Because a complete search is not affordable since the search space is growing exponentially with the number of input words, a repair based algorithm called frobbing is used [2]. The original algorithm is non-incremental as it starts with an analysis of the whole sentence and 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. The generated structure (plus an arbitrary attachment for the new words) is then used as a starting point for the analysis of the extended prefix [7]. This approach is non-monotonic, as the previous PDA provides only a starting point for the next search step. The resulting PDA of each incremental step does not need to include all the arcs of its predecessor and attachments can be changed when encountering input incompatible with the old interpretation. Such a non-monotonic behavior is highly welcome, since it closely resembles the reanalysis strategy observed in the human model.

The grammars for jwcdg are usually manually written, for this work we used a broad coverage grammar for German [2]. jwcdg is able to profit from information contributed by external modules [2]. The hand written constraints are complemented by ones accessing statistical information by those modules. Only the most essential module, a part-of-speech tagger, is used here. The tagger is not run strictly incremental but in a reanalysis mode, re-tagging every new sentence prefix.

Nonspec

Dependency analyses in jwcdg are inclusive, every word must be attached to a head and the system has to assign a head even in the absence of a suitable one. Therefore, a predictive parsing approach is needed for incremental parsing. Both minimal prediction and structural prediction as defined in Section 2 are supported by jwcdg.

The minimal predictive mode is achieved by adding nonspec attachments for every word as possible dependency edges to the search space. To be able to deal with the additional nonspec node, the grammar has to be changed in two regards. First, a constraint is added to slightly penalize nonspec attachments. This guarantees that such an attachment is only chosen if no suitable other head is available. A second change adds guards to the constraints to prevent non-existing attributes of nonspec to 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).

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, e.g. in the case of 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 head itself could be integrated into the rest of the dependency structure without violating additional constraints.

Virtual Nodes

jwcdg also supports a structurally predictive parsing mode where the concept of virtual nodes as defined in Section 2 has been implemented. Since the frobbing search algorithm is not able to add or remove words to or from the constraint optimization problem, a maximal set of potentially useful predictive nodes has to be introduced prior to search.

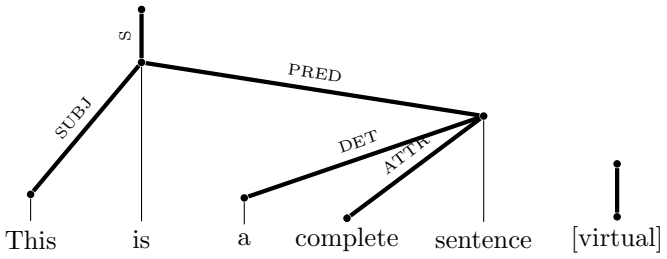
These additional nodes might not all be needed to complete a prefix and unused Virtual Nodes simply stay unconnected. A VN is considered unused if it is assigned to the root node with the empty label as dependency type. In such cases it is not considered part of the sentence and can safely be removed from an analysis without altering its meaning. No other words may be attached to an unused VN. This is enforced by a hard constraint in the grammar. Examples for this are shown in Figure 5.

Virtual nodes, once added to the constraint optimization 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. As with nonspec, attachments to and from virtual nodes are penalized slightly. 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.

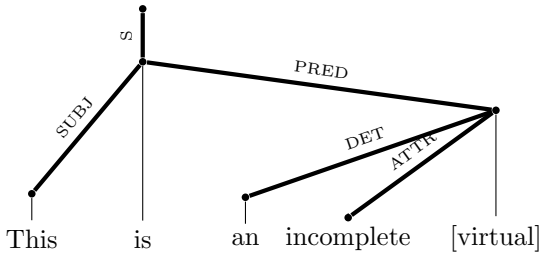
Two kinds of prediction, bottom-up and top-down, can be observed to occur driven by constraint violations. In bottom-up prediction, the inclusion of a predictive node is driven by an unconnected word for which every other integration would result in constraint violations. Top-down prediction is conflict 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 changes to 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.

3.2. MaltParser

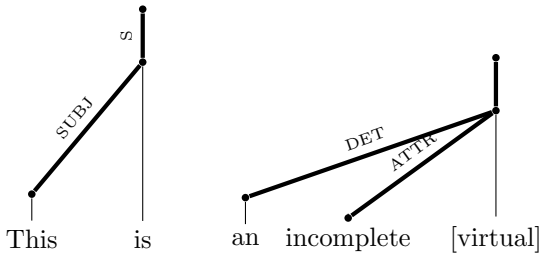
MaltParser [3] is a data driven dependency parser based on a shift-reduce algorithm. It provides a monotonic incremental algorithm but no incremental output in the sense of intermediate analyses for prefixes. We, therefore, modified the parser in a way that allows us to extract partial dependency analyses 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 in a shift action. This allows us to recover the PDAs for every increment.



(a) unused virtual node



(b) used virtual node



(c) a partially used virtual node, a constellation that is not allowed by the grammar

Figure 5. Examples for virtual nodes in the output of jwcdg

The PDAs extracted this way are generally not inclusive as the production of dependency edges is delayed until both ends are available. However, unattached words can be interpreted as attached to nonspec if only those words that have no regent available yet stay unattached. This is provided if using an arc-eager algorithm and explicit root handling.

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. With arc-eager variants of the shift reduce

algorithm, the *right-reduce* action is split into a *right-arc* and a *reduce* action [8]. This modification allows an immediate attachment once both 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 node. 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 turns out not to be exhaustive. Especially punctuation tokens are always left unattached. As they should always attached to the root node, 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 a root attachment would often be incorrect as well.

Given these requirements there are three suitable algorithms: Covington [9], Nivre with pseudo projectivity [10] and 2-Planar [11]. All of them are able to deal with non-projective structures and provide variants for arc-eager processing and explicit root handling. We have chosen the 2-planar algorithm, 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 [11].

3.3. Differences between jwcdg and MaltParser

As should be obvious by now, MaltParser and jwcdg differ in most aspects. While MaltParser is trained on a tree-bank, jwcdg uses a manually generated dependency grammar.

Both parsers apply an incremental algorithm in the sense that information from a previous analysis step are used to calculate the analysis for the extended prefix, but apply different strategies to deal with temporary ambiguity as defined in [12]. While jwcdg applies reanalysis, resulting in timely but non-monotonic output, MaltParser applies lookahead, resulting in monotonic but delayed output. With regard to prediction, jwcdg can either produce structural or minimal prediction while MaltParser can only do the latter.

4. Evaluating Incremental Parser Output

To be able to evaluate incremental parsing results, we need means to quantify and measure the quality of PDAs. There are several difficulties in applying measures for complete sentences to partial dependency analyses. First, there are no gold standard annotated corpora available for sentence prefixes, only for complete sentences. Second, the particular characteristics of incremental output like the

development over time are not captured by evaluating the accuracy of individual PDAs.

4.1. Gold Annotations

There are two ways to obtain gold standard PDAs: either by annotating a corpus of sentence prefixes manually or by systematically generating prefix annotations from existing full sentence annotations. In the first approach, human annotators are presented with a sentence prefix to annotate, without knowledge about how the sentence continues. In the second approach, full sentence annotations are pruned into PDAs.

Due to temporary ambiguity, the syntactic annotation compatible with the complete sentence (obtained by pruning) might not be the most plausible one if looking only at the prefix. Due to pruning, there could even be several different annotations for the same prefix as two sentences might share a prefix but assign a different syntactic structure to the words in the prefix.[3] The resulting corpus would then contain two "correct" analyses for the same sequence of words. This might seem to be a strong argument in favor of manual annotation. However, in the absence of a disambiguating context, annotator choices can be highly influenced by priming effects, resulting in low inter-annotator agreement on what the most plausible annotation for a temporary ambiguous prefix actually is.

One could also assume that the most plausible analysis of a prefix is the one that is most often used in a language. So, while different annotations for the same or similar prefix might occur in a corpus, the more plausible one would still be more common, given a large enough corpus. Therefore, evaluation against such a corpus would still provide the highest score to the parser choosing the most probable interpretation, even though for some prefixes that dependency structure is deemed wrong. Generating prefixes has the advantage that a distribution of prefix annotations is generated. This information is lost in a hand-annotated corpus.

Another source for low inter-annotator agreement is the question how complete the gold standard PDAs should be. As a last point to consider, annotating prefixes requires a multiplied effort compared to annotating complete sentences as there are multiple prefixes for every sentence. For these reasons, the approach of systematic generation from existing annotations is used here.

There are different possibilities of how much prediction should be included in gold standard annotations for sentence prefixes. In [4], two different approaches were investigated, the minimal and the maximal approach. In both cases, the in-prefix words and dependency edges between them (category (a) in Figure 1) appear in the generated gold standard PDA, but the approaches differ in how they handle cases (b)-(d).

In the minimal approach, the gold standard PDAs were generated by ignoring all out-of-prefix words from the full sentence annotation. Attachments to out-of-prefix words are replaced by attachments to nonspec, i.e., all category (b) edges

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

are underspecified. Category (c) and (d) edges, i.e. those with out-of-prefix words as dependents, are discarded. While evaluation against these minimal gold standard PDAs is quite straightforward, they are unsuited to evaluate the predictive part of a PDA.

In the maximal approach, in contrast, the gold standard PDA simply equals the full sentence gold annotation. All words and edges in the structure are treated as potentially predictable. This, however, is an unrealistic goal which will most likely result in a very low prediction recall.

More realistic gold standard PDAs with a prediction level in between the minimal and the maximal ones can be generated based on an estimation of how many out-of-prefix words are actually needed for the dependency structure of sentence prefixes to fulfill the criteria of connectedness and valency saturation. Such an evaluation has been carried out previously[13]. The rules used to determine which words should be predicted can be used to generate gold standard PDAs. Out-of-prefix words are considered necessary if they were part of a connection path between any two in-prefix words (ignoring dependency direction). These connection paths are unambiguous as the dependency structures are cycle free. Also required are all words that fill obligatory valencies of their head, e.g. the subject of a finite verb or the kernel nouns of prepositional phrase.

A word from the full sentence annotation appears in the prefix annotation, if it is

1. an in-prefix word,
2. the (transitive) head of an in-prefix word
3. attached to a in-prefix word via a dependency type indicating an obligatory dependent[4]

Furthermore, a chain of connected out-of-prefix words is folded into one node if they are connected by chain-able dependency types like AUX(illary verbs). Such repeated structures would not be predicted, neither by a human nor a machine parser. An example for this is given in Figure 6.

Annotations created along these guidelines are called **sentence predictive**. Corresponding to the two definitions of connectedness in Section 2, there is a variant of rule 2. Instead of connecting transitively up to the sentence root, only the heads up to the first common head of all in-prefix words are kept. That head is then itself attached to nonspec in the generated gold standard PDA and the resulting annotation is called **fragment predictive**.

In summary, four different gold standard PDA variants can be generated for a prefix: minimal, maximal, sentence predictive and fragment predictive ones. An example with all variants for a single sentence prefix is shown in Figure 7.

4.2. Quality Metrics

The quality of dependency analyses for complete sentences is usually measured by its attachment score (AS). It is defined as the number of words in a sentence

[4]This is highly dependent on the annotation scheme used. According to the annotation scheme used here [14] the following labels are considered as possibly indicating an obligatory valency: SUBJ', 'SUBJC', 'PN', 'CJ', 'OBJA', 'OBJD', 'OBJC', 'PRED','OBJP', 'AUX', 'PART','S'

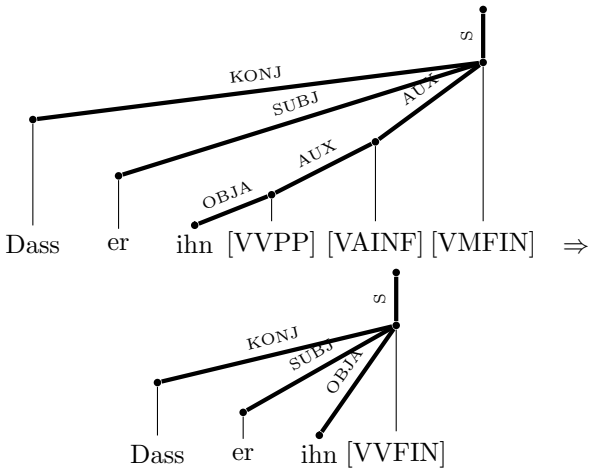


Figure 6. Example for folding: A PDA for the first three words of the German clause "dass er ihn gesehen haben soll" (literally: "that he him saw has should" meaning:"that he is supposed to have seen him"); all three verbs would be kept for connectedness, but are then folded into a single virtual verb

that have been assigned to the same head as in a gold standard annotation of the same sentence (unlabeled attachment score, UAS) and, optionally, with the same label (LAS). This is not directly applicable to PDAs. Firstly, attachments of virtual nodes cannot be trivially deemed correct or incorrect given the gold annotation. Second, PDAs of the same sentence prefix cannot be compared by their attachment score, if they differ in the number of attachments they contain. Furthermore, incremental parser output consists not of a single dependency structure, but a sequence of them. This dynamic is not captured by summing up the attachment score for all individual elements of this sequence.

4.2.1. Correctness of Individual Attachments

To evaluate the quality of a PDA, the correctness of individual attachments has to be determined. Edges between words in the prefix (category (a) edges in Figure 1) can be directly compared against the gold annotation for the complete sentence. In minimally predictive PDAs, words can also be attached to nonspec, forming a category (b) edge. These edges can be compared against the attachment of the same dependent word in the gold annotation from the minimal approach. The attachment of word w to nonspec is correct iff the head of w in the gold standard PDA is also nonspec (which is the case if its head in the full sentence annotation is outside of the current prefix).

Structurally predictive PDAs can be evaluated against minimal gold standard PDAs by treating attachments to virtual nodes as attachments to nonspec and ignoring all (c) and (d) edges. This allows for a direct comparison between PDAs of different prediction levels but ignores the additional information provided by structural prediction. That part can be captured in the evaluation by using structurally predictive gold standard PDAs. When both the tested PDA and the gold standard PDA contain nodes not grounded in the know words, it is, however, not necessarily clear which elements correspond to each other and a mapping between

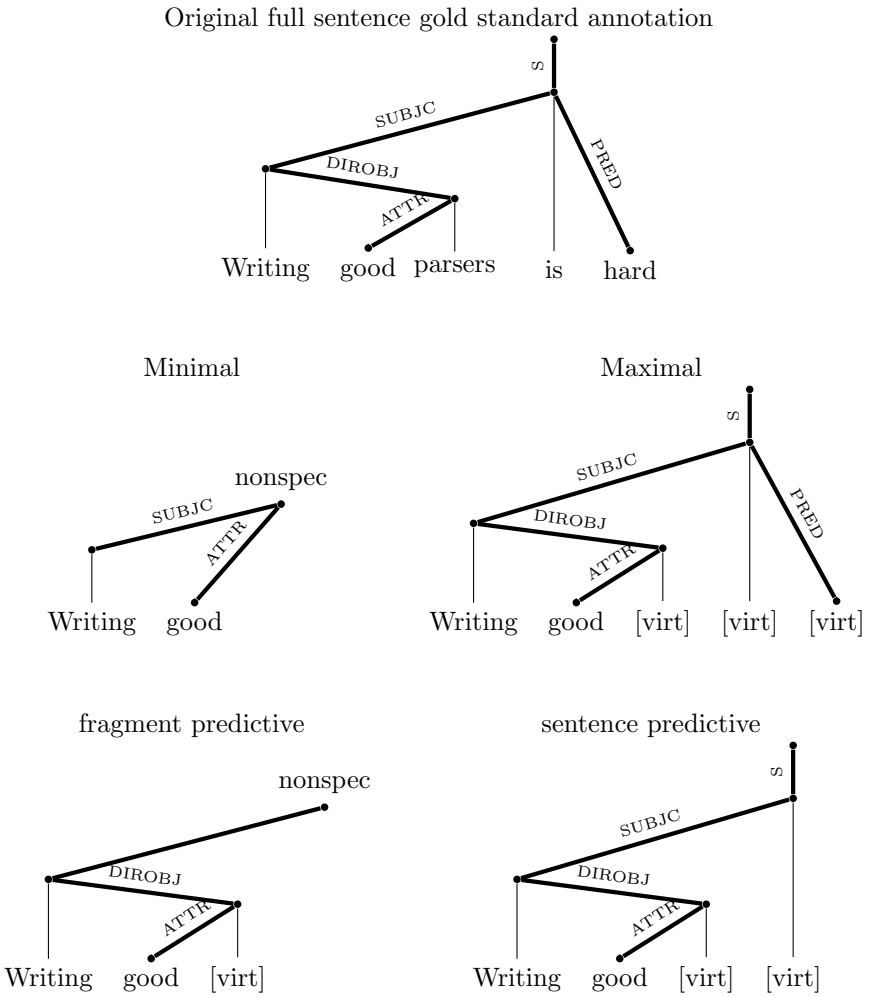


Figure 7. Different levels of gold annotation for the prefix "Writing good" of the sentence "Writing good parsers is hard"

virtual nodes is needed. As several mappings might be possible, the attachment score of a PDA is defined as the maximum over all mappings. This optimum might depend on whether accuracy is measured labeled or unlabeled. Given that a virtual node can partake in more than one dependency edge, i.e. it has one head 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. Since 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. By definition, all predictions still left in the final result are incorrect.

The different prediction levels are not directly comparable as they differ not only in the number of words and attachments per PDA, but also in the level of information carried by an attachment to a predicted head. Two words attached to nonspec might be correct, the same two words attached to the same VN would be incorrect if they are attached to different words in the full sentence annotation. Another differentiation of prediction level is whether a dependency label is assigned to a nonspec attachment. Counting no assigned label as being correct but requiring an assigned label to match the one in the gold annotation is not an adequate evaluation. Output differing in this regard can therefore either only be evaluated unlabeled, or the label has to be ignored selectively for nonspec attachments.

4.2.2. Capturing the Dynamics of Quality

While the quality of individual PDAs has been discussed in the previous section, 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 would be counted more often and, therefore, would have a greater influence on the overall score than words appearing later in the sentence. Also, most of these occurrences would be long after the word first appeared so that an initial wrong attachment repaired later on would be concealed by the many later occurrences. If one, in contrast, considers only the attachment of a word for the prefix it first appears in, phenomena like prediction, delayed attachment or non-monotonicity are neglected. All these appear in later prefix versions or, in the case of prediction, even in earlier ones.

We apply a sliding window to the sequences of PDAs to provide all words with the same degree of importance on the evaluation results while investigating the temporal evolution their attachments. 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. Instead of one accuracy score, a window has n slots holding an accuracy score. For every prefix the attachment of the last word in the prefix contributes to slot #0, the second to last word to slot #1 etc. For n significantly smaller than the average length of sentences, approximately the same weight is given to all words. Words at the end of a sentence still have a slightly smaller weight.

Other measures besides accuracy can be determined for a slot, namely how many words are attached to a predicted head, how many to an in-prefix one and how many have no head assigned at all. Delayed output due to lookahead results in unattached words. This reflects in the window in that for the first m time-points after a word appears in the prefix, it is simply not included in the analysis, where m is the fixed lookahead size. Note, however, that with lookahead there is not a simple one-to-one mapping between PDAs and input increments: As the lookahead window (not to be confused with the sliding window discussed here) must first be filled before any output can be generated, no explicit output will be produced for the first m input increments. Similarly, as the final input increment fills the lookahead window of the last $m + 1$ words, there are actually $m + 1$ output increments for it. This can be compensated by adding m empty

output increments at the beginning and by keeping only the final analysis for the last input increment.

Stability

To be able capture the non-monotonicity of a reanalyzing parser like jwcdg, a **stability** value is determined. The stability score for a given slot is calculated just like the accuracy but the PDA is compared to the final annotation found by the parser instead of the gold standard annotation. So, instead of the percentage of words attached to the correct head, stability gives the percentage of attachments not changed in the final output. For example, a stability of 0.7 for slot #1 means that in 70% of the prefixes the second most recent word was attached to the same head as in the final analysis for the complete sentence. Note that changes in a later PDA are not reflected here, as long as they are changed back in the final dependency analysis.

4.3. Connectedness

Predictive incremental parsing aims at fully connected PDAs where each new word is integrated into the already existing structure. We quantify the degree of disconnectedness of a PDA by means of its average **fragmentation**, 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. In the annotation scheme used for our evaluation [14], punctuation marks are never attached to another word and are, therefore, not considered here. Note that even in the gold standard there are some fragmented annotations.

5. Evaluation

After two incremental parsers have been described in Section 3 and a way to generate gold standard PDAs from whole sentence annotations has been introduced in Section 4, in this Section the output of the parsers on concrete input is evaluated.

5.1. Setup and Data

Evaluation has been carried out on 500 German sentences from a version of Negra corpus [15] converted to dependency structures [16]. Three different gold annotation were generated for every prefix of each of these sentences. The three different levels of gold annotations are minimally predictive, structurally predictive connecting to the sentence root and structurally predictive as a fragment, as described in Section 4.1.

MaltParser was trained on 15000 different sentences from the same corpus. All configurations employ the 2-planar algorithm, the feature set is the one for

German used by Rodriguez[11]. The original feature set uses a lookahead size of three, i.e., features of the next three words in the input buffer are used. Variants for different lookahead sizes down to no lookahead were generated by deleting the respective features.

Both parsers were paired with incremental part of speech taggers as using gold tags or non-incrementally tagged input data would not be true to the use-case of incremental processing. Some strategy to deal with temporary ambiguity is needed to provide good accuracy in incremental PoS-tagging [12]. Unfortunately, the parsers, differ in how they integrate part of speech taggers and are not compatible with the same strategy for incremental PoS tagging. While MaltParser is restricted to monotonic tagger output and can only utilize one tag per word, leaving only lookahead as a strategy, jwcdg is able to integrate multi-tagging and non-monotonic tagger output.

According to a previous evaluation of incremental PoS taggers [12], TnT [17] and SVMTool [18] are suitable candidates. TnT, however, does not provide a lookahead strategy, making it a poor match for MaltParser, while SVMTool does not produce confidence scores that are useful for jwcdg. Therefore, we have chosen TnT with multi-tagging and re-tagging of each prefix for jwcdg, while MaltParser is combined with SVMTool using a lookahead of one or zero, depending on the evaluation run.

Combining MaltParser with a tagger, the total lookahead of such a pipeline amounts to the sum of the individual lookahead for both components. We have investigated configurations with a total lookahead between zero and four. For Incremental jwcdg, three different configurations were used, differing in which extra nodes are available, and changes to the constraints that define how they may be integrated. The first configuration (jwcdg-NS) utilizes nonspec for minimal prediction. The second one (jwcdg-VN) employs virtual nodes for sentence prediction. The third one (jwcdg-VN-NS) uses both types of nodes for fragment prediction. The latter two configurations use a set of one virtual verb and two virtual nouns. This has been shown to be enough to yield a high theoretical coverage for parsing German with structural prediction[13].

All configurations are evaluated against the minimal gold standard. In addition, jwcdg-VN and jwcdg-VN-NS are evaluated against the respective structurally predictive gold standard.

5.2. Discussion

The results in Table 1 show that all three jwcdg configurations result in a similar final accuracy of 87%/85% (U)AS which is around 2% higher than the best MaltParser configuration (la2+1). This gap might be reduced by more training data and an optimized feature set, but optimizing MaltParser is not the main focus here.

Since MaltParser is not able to produce labels for attachments to nonspec, all such attachments are incorrect in the labeled evaluations. Therefore, we introduce a third variant (**semi-labeled**) which is the same as labeled but disregards the labels of edges to nonspec. This variant allows for a labeled evaluation where the minimal predictive attachments are maximally unspecified.

Table 1. Evaluation of MaltParser and jwcdg with different configurations regarding prediction and lookahead; Lookahead numbers are given as "parser LA + tagger LA"; Initial AS: The attachment score for the first attachment decision a parser makes for a word (note that this decision is delayed by lookahead); Fragm.: The average fragmentation of the resulting PDAs.

		initial AS			final AS		
Parser	Conf.	unlabeled	labeled	semi-labeled	unlabeled	labeled	Fragm.
Minimal Prediction:							
jwcdg	NS	72.44%	58.31%	71.04%	87.19%	84.75%	1.058
	VNs	78.93%	73.38%	77.40%	87.18%	84.91%	0.055
	VN-NS	77.21%	65.89%	75.64%	87.45%	85.21%	0.048
Malt	LA 0+0	85.83%	46.57%	84.15%	83.05%	79.18%	1.453
	LA 1+0	88.11%	48.21%	86.49%	85.57%	81.79%	1.392
	LA 2+0	88.27%	48.33%	86.57%	85.55%	81.48%	1.401
	LA 2+1	89.21%	49.58%	86.48%	86.51%	82.91%	1.274
	LA 3+1	88.83%	49.43%	87.27%	86.03%	82.57%	1.279
Structural Prediction:							
jwcdg	VNs	76.43%	71.18%	74.90%	87.18%	84.91%	0.055
	VN-NS	75.65%	65.01%	67.28%	87.45%	85.21%	0.048

Table 2. Precision and Recall of the Virtual Nodes in structurally predictive PDAs. Note that two different variants of the gold annotations were used

		unlabeled		labeled	
Parser configuration	gold standard used	precision	recall	precision	recall
jwcdg VNs	sentence prediction	45.21%	49.18%	36.18%	39.19%
jwcdg VN-NS	fragment prediction	40.66%	44.04%	30.47%	33.00%

Figures 8 to 12 show the temporal evolution of accuracy and stability within the evaluation window for different parser configurations. In these plots, the differences in the behavior of MaltParser and jwcdg 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, possibly changing a correct prediction into a wrong attachment. 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 is the case in jwcdg. Here, the labeled stability of initial attachments is only 75%, i.e., 25% of the attachments had to be changed later on. [5]

The delayed output of the configurations with lookahead leads to a smaller number of slots in the evaluation window having received any attachments. The few assignments in slots 0-3 in Figure 12 are due to an end-of-sentence effect where the lookahead window is filled preliminarily.

A noteworthy observation is that jwcdg produces significantly more erroneous initial attachments than MaltParser. Also the proportion between the attach-

[5] 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 a stability reduction of 5%

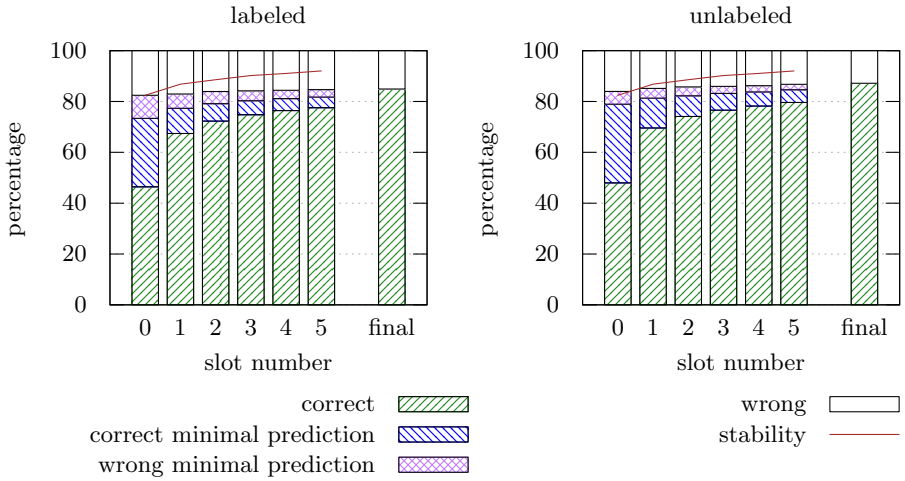


Figure 8. Scores of jwcdg with virtual nodes for a sliding window with 6 slots; slot 0 refers to the most recent word in the input, slot 1 to the second most recent and so on.

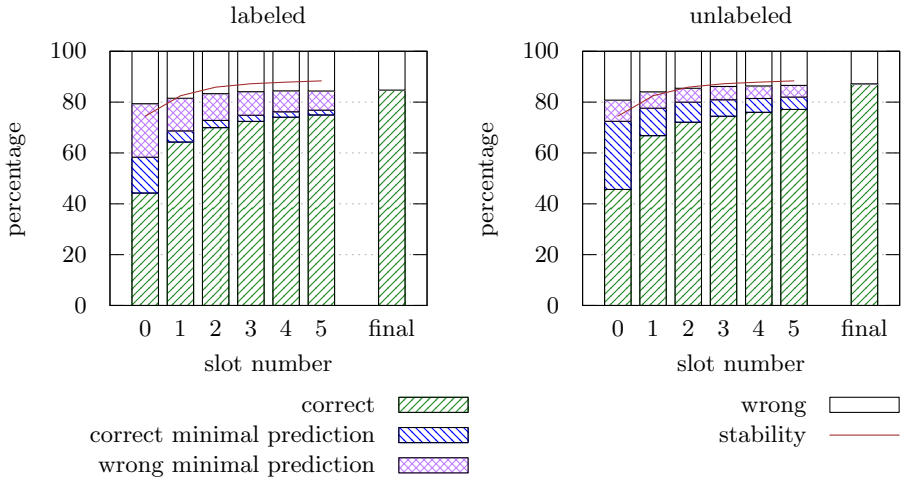


Figure 9. Scores of jwcdg with nonspec for a sliding window with 6 slots; slot 0 refers to the most recent word in the input, slot 1 to the second most recent and so on.

ments to in-prefix words and predictive attachments is different, indicating that many of the wrong attachments to in-prefix words should have been directed out of the prefix. Obviously, jwcdg is too eager to attach words to already available heads but by means of reanalysis it is able to recover in many cases. This difference of better initial attachment in MaltParser and the better final accuracy and reanalysis capability of jwcdg suggests that a combination of the two

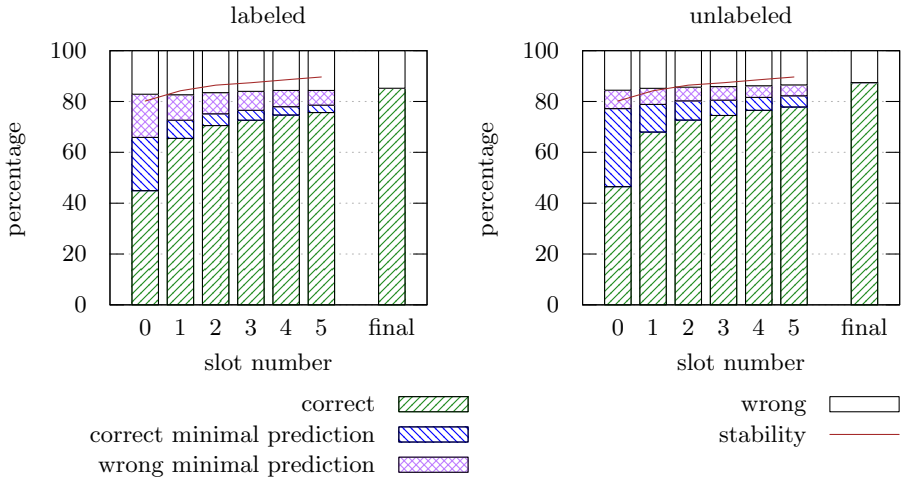


Figure 10. Scores jwcdg using both VNs and nonspec for a sliding window with 6 slots; slot 0 refers to the most recent word in the input, slot 1 to the second most recent and so on.

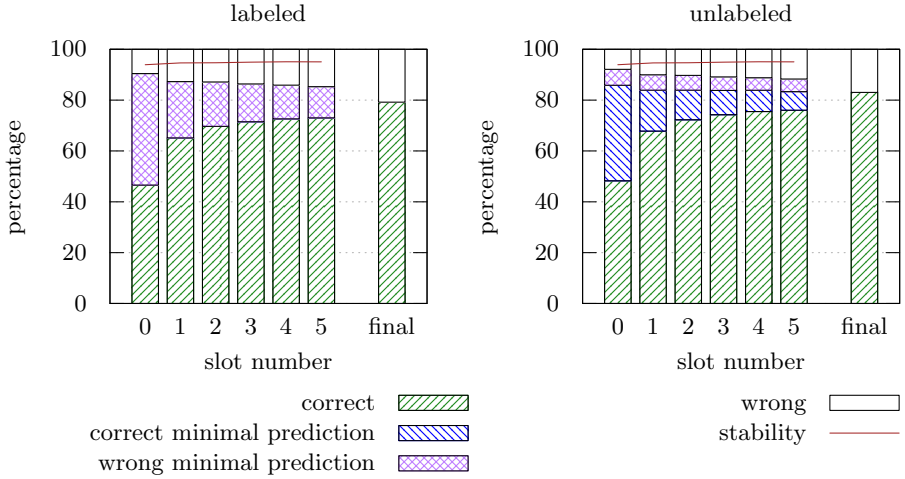


Figure 11. Scores of MaltParser with no lookahead for a sliding window with 6 slots; slot 0 refers to the most recent word in the input, slot 1 to the second most recent and so on.

parsers would be beneficial to increase PDA quality. Recent results support this supposition [19].

When compared against the minimal gold standard, the jwcdg configurations using virtual nodes fare better than the one using only nonspec, even though the additional information they provide is not honored by that evaluation: The assignment of labels for predictive attachments is improved as is the quality of

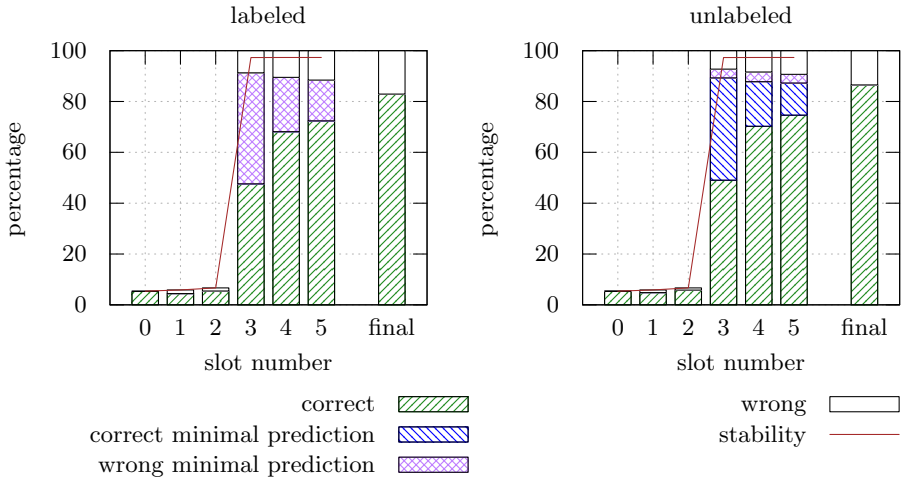


Figure 12. Scores of MaltParser with lookahead 3 for a sliding window with 6 slots; slot 0 refers to the most recent word in the input, slot 1 to the second most recent and so on.

attachments between words in the prefix. This indicates that virtual nodes help jwcdg to find better structures overall.

As expected, the initial attachment is slightly lower when comparing against one of the predictive gold standards because the definition of a correct attachment is more harsh: an attachment to a VN is only correct if it is mapped to the correct VN in the gold standard. The final accuracy is the same, as there are no predictive attachments in final PDAs.

As seen in Table 2, nearly half of the virtual nodes in the gold standard were correctly predicted by jwcdg. The precision is also around 45%, indicating that jwcdg does not predict too few VNS but around half of the attachments are incorrect. This is not surprising, since there might be more than one plausible prediction for a sentence prefix, but only one of them is deemed correct by the gold standard. An interesting finding here is that jwcdg-VN performs better (against the sentence predictive gold standard) than jwcdg-VN-NS (against the fragment predictive gold standard) by around 5% for both precision and recall. A possible explanation for this is that the additional VNs in the sentence predictive gold standard, i.e. mostly the finite verb connected to the sentence root, are easier to predict than other VNs.

An important benefit of structural prediction, though, is the significantly reduced amount of fragmentation, as indeed the PDAs generated by jwcdg VNs are connected to a similar degree as the gold standard annotations: jwcdg's output has an average fragmentation of 0.055, while the sentence prediction gold standard has an average fragmentation of 0.072.

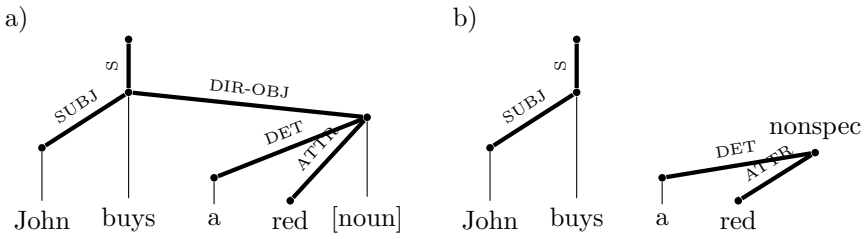


Figure 13. A connected (a) and an unconnected (b) PDA

6. Related Work

To our knowledge, partial dependency analyses have not been investigated previously in detail. Work on incremental dependency parsing like [20] 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 [7], the evolution of partial analyses has been studied, but no broad scale evaluation has been carried out.

An alternative grammar formalism able to produce connected partial analyses is PLTAG [21], a variant of the Tree Adjoining Grammar (TAG) formalism. In this approach, prediction plays a strong role, too. Top-down prediction is facilitated through substitution nodes in lexical entries, e.g. verb 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 TAG is beyond the scope of this work but a promising topic for further research.

The metrics presented here 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 is able to 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 13 a) compared to the not connected analysis (13 b)), it would be desirable to assign a higher recall value to a). An application oriented measure for prefix analyses is defined by [22] where several variants for incremental reference resolution are discussed. They are, however, only applicable for utterances with a single reference.

7. Conclusions

We presented a definition of partial dependency analyses that allows us to derive fully connected structures for sentence prefixes by introducing predicted nodes into the dependency graph. PDAs provide the possibility to encode prediction in different amounts and granularities. The customary attachment score metric alone is not sufficient to capture the dynamics of incremental parser output. Therefore, an extended evaluation scheme has been introduced that describes the evolution of accuracy measures over a window of a fixed number of recent words. For this

scheme, several methods to automatically create incrementalized gold standards have been proposed, which are tailored towards specific evaluation settings.

Using these measures, two existing dependency parsers have been compared, jwcdg and MaltParser. While MaltParser is capable of producing better initial attachments (when ignoring labels), jwcdg can produce structural prediction, yields a higher final accuracy and does not need lookahead. Also, jwcdg utilizes reanalysis can therefore revise its output if new information is available, resembling human parsing strategies. Using virtual nodes instead of nonspec when parsing with jwcdg does not only result in richer prediction but also helps jwcdg finding better attachments in the prefixes.

References

[1] Patrick Sturt and Vincent Lombardo. Processing coordinated structures: Incrementality and connectedness. *Cognitive Science*, 29:291–305, 2005.

[2] Kilian A. Foth. *Hybrid Methods of Natural Language Analysis*. PhD thesis, Uni Hamburg, 2006.

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

[4] Niels Beuck, Arne Köhn, and Wolfgang Menzel. Incremental Parsing and the Evaluation of Partial Dependency Analyses. In *DepLing 2011, Proceedings of the 1st International Conference on Dependency Linguistics*, 2011.

[5] Michael Daum. Dynamic Dependency Parsing. In *In Proceedings of the ACL 2004 Workshop on Incremental Parsing*, 2004.

[6] Ingo Schröder. *Natural Language Parsing with Graded Constraints*. PhD thesis, Uni Hamburg, 2002.

[7] Wolfgang Menzel. Towards radically incremental parsing of natural language. In *Recent Advances in Natural Language Processing V*, 2009.

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

[9] Michael A. Covington. A fundamental algorithm for dependency parsing. In *In Proceedings of the 39th Annual ACM Southeast Conference*, pages 95–102, 2001.

[10] Joakim Nivre. Dependency Grammar and Dependency Parsing. Technical report, Växjö University: School of Mathematics and Systems Engineering, 2005.

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

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

[13] Niels Beuck and Wolfgang Menzel. Structural Prediction in Incremental Dependency Parsing. In Alexander Gelbukh, editor, *Computational Linguistics and Intelligent Text Processing*, volume 7816 of *Lecture Notes in Computer Science*, pages 245–257. Springer Berlin Heidelberg, 2013.

[14] Kilian Foth. Eine umfassende Constraint-Dependenz-Grammatik des Deutschen. Technical report, Universität Hamburg, 2006.

[15] Thorsten Brants, Roland Hendriks, Sabine Kramp, Brigitte Krenn, Cordula Preis, Wojciech Skut, and Hans Uszkoreit. Das NEGRA-Annotationsschema. Negra project report, Universität des Saarlandes, Computerlinguistik, Saarbrücken, Germany, 1997.

[16] Michael Daum, Kilian Foth, and Wolfgang Menzel. Automatic transformation of phrase treebanks to dependency trees. In *4th Int. Conf. on Language Resources and Evaluation, LREC-2004*, 2004.

[17] Thorsten Brants. TnT - A Statistical Part-of-Speech Tagger. In *ANLC 00 Proceedings of the sixth conference on Applied natural language processing*, 2000.

[18] Jesús Giménez and Lluís Màrquez. SVMtool: A general POS tagger generator based on Support Vector Machines. In *Proceedings of the 4th LREC*, 2004.

[19] Arne Köhn and Wolfgang Menzel. Incremental and Predictive Dependency Parsing under Real-Time Conditions. In *Proceedings of RANLP 2013*, 2013. to appear.

[20] Joakim Nivre. Incrementality in Deterministic Dependency Parsing. In *Incremental Parsing: Bringing Engineering and Cognition Together, Workshop at ACL 2004*, 2004.

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

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