Incremental Predictive Parsing with TurboParser

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

Abstract

Most approaches to incremental parsing either incur a degradation of accuracy or they have to postpone decisions, yielding underspecified intermediate output. We present an incremental predictive dependency parser that is fast, accurate, and largely language independent. By extending a state-of-the-art dependency parser, connected analyses for sentence prefixes are obtained, which even predict properties and the structural embedding of upcoming words. In contrast to other approaches, accuracy for complete sentence analyses does not decrease.

1 Introduction

When humans communicate by means of a natural language, utterances are not produced at once but evolve over time. Human interaction benefits from this property by processing yet unfinished utterances and reacting on them. Computational parsing on the other hand is mostly performed on complete sentences, a processing mode which renders a responsive interaction based on incomplete utterances impossible.

When spoken language is analyzed, a mismatch between speech recognition and parsing occurs: If parsing does not work incrementally, the overall system loses all the desirable properties made possible by incremental processing. For speech dialogue systems, this leads to increased reaction times and an unnatural ping-pong style of interaction (Schlangen and Skantze, 2011).

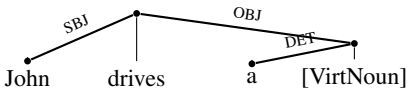
1.1 Desirable features of incremental parsers

Dependency parsing assigns a head and a dependency label to each word form of an input sentence and the resulting analysis of the sentence is usually required to form a tree. An incremental dependency parser processes a sentence word by word, building analyses for sentence prefixes (*partial dependency analyses, PDA*), which are extended and modified in a piecemeal fashion as more words become available.

A PDA should come with three important (but partly contradictory) properties: beyond being accurate, it should also be as stable and informative as possible. Stability can be measured as the amount of structure (attachments and their labels) of a PDA a_i which is also part of the analysis a_n of the whole sentence. To be maximally informative, at least all available word forms should be integrated into the prefix PDA. Even such a simple requirement cannot easily be met without predicting a structural skeleton for the word forms in the upcoming part of the sentence(bottom-up prediction). Other predictions merely serve to satisfy completeness conditions (i.e. valency requirements) in an anticipatory way (top-down predictions). In fact, humans are able to derive such predictions and they do so during sentence comprehension (Sturt and Lombardo, 2005).

Without prediction, the sentence prefix “John drives a” of “John drives a car” can only be parsed as a disconnected structure:

The determiner remains unconnected to the rest of the sentence, because a possible head is not yet available. However, the determiner could be integrated into the PDA if the connection is established by means of a predicted word form, which has not yet been observed. Beuck et al. (2011) propose to use *virtual nodes* (VNs) for this purpose. Each VN represents exactly one upcoming word. Its lexical instantiation and its exact position remain unspecified. Using a VN, the prefix “John drives a” could then be parsed as follows, creating a fully connected analysis, which also satisfies the valency requirements of the finite verb.



This analysis is clearly more informative but still restricted to the existence of a noun filling the object role of "drives" without predicting its position. Although a VN does not specify the lexical identity of the word form it represents, it can nonetheless carry some information such as a coarse-grained part-of-speech category.

1.2 Related work

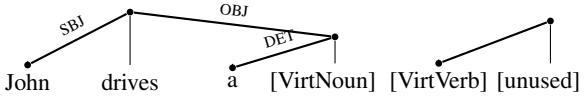
Parsers that produce incremental output are relatively rare: PLTag (Demberg-Winterfors, 2010) aims at psycholinguistic plausibility. It makes tradeoffs in the field of accuracy and coverage (they report 6.2 percent of unparseable sentences on sentences of the Penn Treebank with less than 40 words). Due to its use of beam search, the incremental results are non-monotonic. Hassan et al. (2009) present a CCG-based parser that can parse in an incremental mode. The parser guarantees that every parse of an increment extends the previous parse monotonically. However, using the incremental mode without look-ahead, parsing accuracy drops from 86.70% to 59.01%. Obviously, insisting on strict monotonicity ($a_i \subseteq a_n$) is too strong a requirement, since it forces the parser to keep attachments that later turn out to be clearly wrong in light of new evidence.

Being a transition-based parser, Maltparser (Nivre et al., 2007) does incremental parsing by design. It is, however, not able to predict upcoming structure and therefore its incremental output is usually fragmented into several trees. In addition, Maltparser needs a sufficiently large look-ahead to achieve high accuracy (Beuck et al., 2011).

Beuck et al. (2011) introduced incremental and predictive parsing using Weighted Constraint Dependency Grammar. While their approach does not decrease in incremental mode, it is much slower than most other parsers. Another disadvantage is its hand-written grammar which prevents the parser from being adapted to additional languages by simply training it on an annotated corpus and which makes it difficult to derive empirically valid conclusions from the experimental results.

2 Challenges for predictive parsing

Extending a dependency parser to incremental parsing with VNs introduces a significant shift in the problem to be solved: While originally the problem was *where* to attach each word to (1), in the incremental case the additional problem arises, *which* VNs to include into the analysis (2). Problem (2), however, depends on the syntactic structure of the sentence prefix. Therefore, it is not possible to determine the VNs *before* parsing commences, but the decision has to be made *while* parsing is going on. We can resolve this issue by transforming problem (2) into problem (1) by providing the parser with an additional node, named *unused*. It is always attached to the special node 0 (the root node of every analysis) and it can only dominate VNs. *unused* and every VN it dominates are not considered part of the analysis. Using this idea, the problem of whether a VN should be included into the analysis is now reduced to the problem of where to attach that VN:



To enable the parser to include VNs into PDAs, a set of VNs has to be provided. While this set could include any number of VNs, we only include a set that covers most cases of prediction since rare virtual nodes have a very low a-priori probability of being included and additional VNs make the parsing problem more complex. This set is language-dependent and has to be determined in advance. It can be obtained by generating PDAs from a treebank and counting the occurrences of VNs in them. Eventually, a set of VNs is used that is a super-set of a large enough percentage ($> 90\%$) of the observed sets.

3 Gold annotations for sentence prefixes

Annotating sentence prefixes by hand is prohibitively costly because the number of increments is a multitude of the number of sentences in the corpus. Beuck and Menzel (2013) propose an approach to automatically generate predictive dependency analyses from the annotation of full sentences. Their method tries to generate upper bounds for predictability which are relatively tight. Therefore, not everything that is deemed predictable by the algorithm is predictable in reality, but everything that is predictable should be deemed as predictable: Let W be all tokens of the sentence and P the set of tokens that lie in the prefix for which an incremental analysis should be generated. A word $w \in W \setminus P$ is assumed to be predictable ($w \in Pr$) if one of the following three criteria is met:

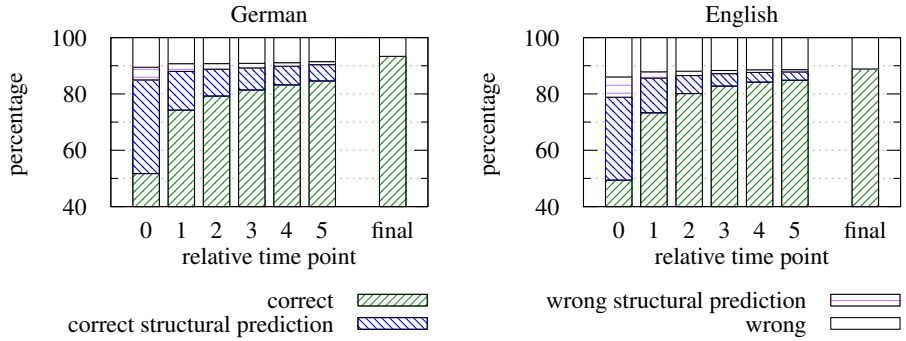
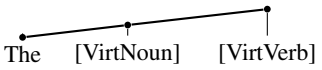


Figure 1: Results for TurboParser for German and English with gold standard PoS (labeled)

bottom-up prediction w lies on the path from some $w' \in P$ to 0. E. g., given the sentence prefix "The", an upcoming noun and a verb is predicted:



top down prediction $\pi(w)$, the head of w, is in $P \cup Pr$, and w fills a syntactic role – encoded by its dependency label – that is *structurally determined*. That means w can be predicted independently of the lexical identity of $\pi(w)$. An example for this is the subject label: If $\pi(w)$ is in Pr and w is its subject, w is assumed to be predictable.

lexical top-down prediction $\pi(w) \in P$ and w fills a syntactic role that is determined by an already observed lexical item, e.g. the object role: If $\pi(w)$ is a known verb and w is its object, $w \in Pr$ because it is required by a valency of the verb.

While this procedure is language-independent, some language-specific transformations must be applied nonetheless. For English, parts of gapping coordinations can be predicted whereas others can not. For German, the transformations described in (Beuck and Menzel, 2013) have been used without further changes. Both sets of structurally and lexically determined roles are language dependent. The label sets for German have been adopted from (Beuck and Menzel, 2013), while the sets for English have been obtained by manually analyzing the PTB (Marcus et al., 1994) for predictability.

For words marked as predictable their existence and word class, but not their lexicalization and position can be predicted. Therefore, we replace the lexical item with "[virtual]" and generalize the part-of-speech tag to a more coarse grained one.

4 Predictive parsing with TurboParser

We adapt TurboParser (Martins et al., 2013) for incremental parsing because it does not impose structural constraints such as single-headedness in its core algorithm. For each parsing problem, it creates an integer linear program – in the form of a factor graph – with the variables representing the possible edges of the analyses.

Since well-formedness is enforced by factors, additional constraints on the shape of analyses can be imposed without changing the core algorithm of the parser. We define three additional restrictions with respect to VNs: 1) A VN that is attached to *unused* may not have any dependents. 2) A VN may not be attached to 0 if it has no dependents. 3) Only VNs may be attached to the *unused* node.

For a given sentence prefix, let A be the set of possible edges, V the set of all vertices, $N \subset V$ the VNs and $u \in V$ the *unused* node. Moreover, let $B \subset A$ be the set of edges building a well-formed analysis and $z_a \triangleq \mathbb{I}(a \in B)$, where $\mathbb{I}(.)$ is the indicator function. The three additional conditions can be expressed as linear constraints which ensure that every output is a valid PDA:

$$z_{\langle n,j\rangle} + z_{\langle u,n\rangle} \leq 1, \qquad n \in N, j \in V \qquad (1)$$

$$z_{\langle 0,n\rangle} \leq \sum_{j \in V} z_{\langle n,j\rangle}, \qquad n \in N \qquad (2)$$

$$z_{\langle u,i\rangle} = 0, \qquad i \in V \setminus N \qquad (3)$$

The current implementation is pseudo-incremental. It reinitializes the ILP for every increment without passing intermediate results from one incremental processing step to the next, although this might be an option for further optimization.

High quality incremental parsing results can not be expected from models which have only been trained on whole-sentence annotations. If a parser is trained on gold-standard PDAs (generated as described in section 3), it would include every VN into every analysis because that data does not include any non-attached VNs. We therefore add non-attached VNs to the generated PDAs until they contain at least the set of VNs that is later used during parsing. For instance, each German training increment contains at least one virtual verb and

two virtual nouns and each English one at least one virtual verb and one virtual noun. This way, the percentage of VNs of a specific type being attached in the training data resembles the a priori probability that a VN of that type should be included by the parser while parsing.

TurboParser is trained on these extended PDAs and no adaptation of the training algorithm is needed. The training data is heavily skewed because words at the beginning of the sentences are more prevalent than the ones at the end. As a comparison with a version trained on non-incremental data shows, this has no noticeable effect on the parsing quality.

5 Evaluation

The usual methods to determine the quality of a dependency parser – labeled and unlabeled attachment scores (AS) – are not sufficient for the evaluation of incremental parsers. If the AS is computed for whole sentences, all incremental output is discarded and not considered at all. If every intermediate PDA is used, words at the start of a sentence are counted more often than the ones at the end. No information becomes available on how the accuracy of attachments evolves while parsing proceeds, and the prediction quality (i.e. the VNs) is completely ignored. Therefore, we adopt the enhanced mode of evaluation proposed by Beuck et al. (2013): In addition to the accuracy for whole sentences, the accuracies of the n newest words of each analysis are computed. This yields a curve that shows how good a word can be assumed to be attached depending on its distance to the most recent word.

Let $\langle V, G\rangle$ be the gold standard analysis of an increment and $\langle V', P\rangle$ the corresponding parser output. V and V' are the vertices and G and P the respective edges of the analyses. Let V'_p and V'_v be the in-prefix and virtual subset of V', respectively. To evaluate the prediction capabilities of a parser, for each increment an optimal partial, surjective mapping[1] $V' \to V$ from the output produced by the parser to the (automatically generated) gold standard is computed, where each non-virtual element of V' has to be mapped to the corresponding element in V. Let M be the set of all such mappings. Then the best mapping is defined as follows:

$$\phi = \arg\max_{m \in M} \sum_{w \in V'} \mathbb{I}(\pi(m(w)) = m(\pi(w)))$$

[1]The mapping is partial because for some VNs in V' there might be no corresponding VN in the gold standard.

We define a word w as correctly attached (ignoring the label) if $\pi(\phi(w)) = \phi(\pi(w))$. In an incremental analysis, an attachment of a word w can be classified into four cases:

correct $\pi(\phi(w)) = \phi(\pi(w))$, $\pi(w) \in V'_p$
corr. pred. $\pi(\phi(w)) = \phi(\pi(w))$, $\pi(w) \in V'_v$
wrong pred. $\pi(\phi(w)) \neq \phi(\pi(w))$, $\pi(w) \in V'_v$
wrong $\pi(\phi(w)) \neq \phi(\pi(w))$, $\pi(w) \in V'_p$

We can count the number of VNs that have been correctly attached: Let T be the set of all analyses produced by the parser and ϕ_t the best mapping as defined above for each $t \in T$. Furthermore, let $vn(t)$ be the set of VNs in t. The total number of correct predictions of VNs is then defined as:

$$corr = \sum_{t \in T} \sum_{v \in vn(t)} \mathbb{I}(\pi(\phi_t(v)) = \phi_t(\pi(v)))$$

Precision and recall for the prediction with VNs can be computed by dividing *corr* by the number of predicted VNs and the number of VNs in the gold standard, respectively.

Evaluation has been carried out on the PTB converted to dependency structure using the LTH converter (Johansson and Nugues, 2007) and on the Hamburg Dependency Treebank (Foth et al., 2014). From both corpora predictive PDAs padded with unused virtual nodes have been created for training. For English, the sentences of part 1-9 of the PTB were used, for German the first 50,000 sentences of the HDT have been selected. Testing was done using one virtual noun and one virtual verb for English and two virtual nouns and one virtual verb for German because these sets cover about 90% of the prefixes in both training sets.

Figure 1 shows the evaluation results for parsing German and English using TurboParser. For both languages the attachment accuracy rises with the amount of context available. The difference between the attachment accuracy of the most recent word (relative time point 0, no word to the right of it) and the second newest word (time point 1) is strongest, especially for English. The word five elements left of the newest word (time point 5) gets attached with an accuracy that is nearly as high as the accuracy for the whole sentence (final).

The types of errors made for German and English are similar. For both German and English the unlabeled precision reaches more than 70% (see Table 1). Even the correct dependency label of upcoming words can be predicted with a fairly high precision. TurboParser parses an increment in about 0.015 seconds, which is much faster than WCDG

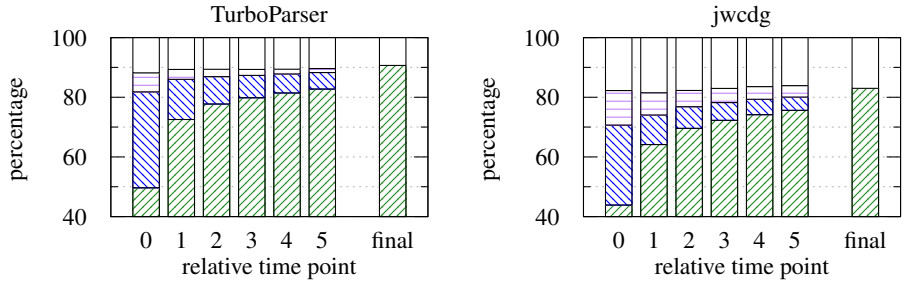


Figure 2: Results for TurboParser and jwcdg for German with tagger (labeled).

	English		German		German&tagger		German (jwcdg)	
	labeled	unlabeled	labeled	unlabeled	labeled	unlabeled	labeled	unlabeled
precision	75.47%	78.55%	67.42%	75.90%	65.21%	73.39%	32.95%	42.23%
recall	57.92%	60,29%	46.77%	52.65%	45.79%	51.54%	35.90%	46.00%

Table 1: Precision and recall for the prediction of virtual nodes

	time point 0		time point 5	
	unlabeled	labeled	unlabeled	labeled
En	89.28%	84.92%	97.32%	97.11%
De	90.91%	88.96%	96.11%	95.65%

Table 2: Stability measures

where about eight seconds per word are needed to achieve a good accuracy (Köhn and Menzel, 2013). The prediction recall is higher for English than for German which could be due to the differences in gold-standard annotation.

Training TurboParser on the non-incremental data sets results in a labeled whole-sentence accuracy of 93.02% for German.The whole-sentence accuracy for parsing with VNs is 93.33%. This shows that the additional mechanism of VNs has no negative effects on the overall parsing quality.

To compare TurboParser and WCDG running both in the predictive incremental mode, we use jwcdg, the current implementation of this approach. jwcdg differs from most other parsers in that it does not act on pre-tagged data but runs an external tagger itself in a multi-tag mode. To compare both systems, TurboParser needs to be run in a tagger-parser pipeline. We have chosen TurboTagger without look-ahead for this purpose. Running TurboParser in this pipeline leads to only slightly worse results compared to the use of gold-standard tags (see Figure 2). TurboParser's attachment accuracy is about ten percentage points better than jwcdg's across the board. In addition, its VN prediction is considerably better.

To measure the stability, let P_i be a prefix of the sentence P_n and a_i and a_n be the corresponding analyses produced by the parser. An attachment of a word $w \in P_i$ is stable if either w's head is the same in a_i and a_n or w's head is not part of P_i in both a_i and a_n. The second part covers the case where the parser predicts the head of w to lie in the future and it really does, according to the final parse. Table 2 shows the attachment stability of the newest word at time point 0 compared to the word five positions to the left of time point 0. TurboParser's stability turns out to be much higher than jwcdg's: For German Beuck et al. (2013) report a stability of only 80% at the most recent word. Interestingly, labeling the newest attachment for English seems to be much harder than for German.

6 Conclusion

Using a parser based on ILP, we were able to analyze sentences incrementally and produce connected dependency analyses at every point in time. The intermediate structures produced by the parser are highly informative, including predictions for properties and structural embeddings of upcoming words. In contrast to previous approaches, we achieve state-of-the-art accuracy for whole sentences by abandoning strong monotonicity and aim at high stability instead, allowing the parser to improve intermediate results in light of new evidence.

The parser is trained on treebank data for whole sentences from which prefix annotations are derived in a fully automatic manner. To guide this process, a specification of structurally and lexically determined dependency relations and some additional heuristics are needed. For parsing, only a set of possible VNs has to be provided. These are the only language specific components required. Therefore, the approach can be ported to other languages with quite modest effort.

References

Niels Beuck and Wolfgang Menzel. 2013. 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.

Niels Beuck, Arne Köhn, and Wolfgang Menzel. 2011. Incremental parsing and the evaluation of partial dependency analyses. In *Proceedings of the 1st International Conference on Dependency Linguistics*. Depling 2011.

Niels Beuck, Arne Köhn, and Wolfgang Menzel. 2013. Predictive incremental parsing and its evaluation. In Kim Gerdes, Eva Hajičová, and Leo Wanner, editors, *Computational Dependency Theory*, volume 258 of *Frontiers in Artificial Intelligence and Applications*, pages 186 – 206. IOS press.

Vera Demberg-Winterfors. 2010. *A Broad-Coverage Model of Prediction in Human Sentence Processing*. Ph.D. thesis, University of Edinburgh.

Kilian A. Foth, Arne Köhn, Niels Beuck, and Wolfgang Menzel. 2014. Because size does matter: The Hamburg Dependency Treebank. In *Proceedings of the Language Resources and Evaluation Conference 2014*. LREC, European Language Resources Association (ELRA).

Hany Hassan, Khalil Sima'an, and Andy Way. 2009. Lexicalized semi-incremental dependency parsing. In *Proceedings of the International Conference RANLP-2009*, pages 128–134, Borovets, Bulgaria, September. Association for Computational Linguistics.

Richard Johansson and Pierre Nugues. 2007. Extended constituent-to-dependency conversion for English. In *Proceedings of NODALIDA 2007*, pages 105–112, Tartu, Estonia, May 25-26.

Arne Köhn and Wolfgang Menzel. 2013. Incremental and predictive dependency parsing under real-time conditions. In *Proceedings of the International Conference Recent Advances in Natural Language Processing RANLP 2013*, pages 373–381, Hissar, Bulgaria, September. INCOMA Ltd. Shoumen, BULGARIA.

Mitchell Marcus, Grace Kim, Mary Ann Marcinkiewicz, Robert MacIntyre, Ann Bies, Mark Ferguson, Karen Katz, and Britta Schasberger. 1994. The Penn Treebank: Annotating predicate argument structure. In *Proceedings of the Workshop on Human Language Technology*, HLT '94, pages 114–119, Stroudsburg, PA, USA. Association for Computational Linguistics.

Andre Martins, Miguel Almeida, and Noah A. Smith. 2013. Turning on the turbo: Fast third-order non-projective turbo parsers. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 617–622, Sofia, Bulgaria, August.

Joakim Nivre, Johan Hall, Jens Nilsson, Atanas Chanev, Gülsen Eryigit, Sandra Kübler, Svetoslav Marinov, and Erwin Marsi. 2007. Maltparser: A language-independent system for data-driven dependency parsing. *Natural Language Engineering*, 13(2):95–135.

Davin Schlangen and Gabriel Skantze. 2011. A general, abstract model of incremental dialogue processing. *Dialogue and Discourse*, 2(1):83–111.

Patrick Sturt and Vincenzo Lombardo. 2005. Processing coordinated structures: Incrementality and connectedness. *Cognitive Science*, 29(2):291–305.