Evaluation of Out-of-Domain Dependency Parsing for its Application in a Digital Humanities Project

Benedikt Adelmann
Wolfgang Menzel
Department of Informatics
Universität Hamburg
adelmann@informatik.uni-hamburg.de
menzel@informatik.uni-hamburg.de

Melanie Andresen
Heike Zinsmeister
Institute for German Language and Literature
Universität Hamburg
melanie.andresen@uni-hamburg.de
heike.zinsmeister@uni-hamburg.de

Abstract

In this paper we evaluate the out-of-domain performance of six commonly used parsers. Our work is situated in a digital humanities project, in which we are interested in the analysis of various text types such as literature or academic text, for which we do not have sufficient training data available. In our evaluation, we focus on those dependency labels that are most relevant to further analysis for research questions in the humanities. The results show good overall performance with Mate being the most successful parser. In general, however, the performance on some labels of interest to humanist scholars (e. g. non-verbal predicates) is rather poor.

1 Introduction

Many digital humanities (DH) projects are interested in the analysis of diverse text types such as literature, spoken text or academic writing, both historic and contemporary. They employ linguistic annotation as pre-processing for further discipline-specific analyses. A common problem is that for most NLP tasks, there is not sufficient annotated training data available for the text types of interest. To annotate such text types automatically, one has to resort to tools trained on available annotated resources, which mostly consist of newspaper text. Applying these tools to the texts of interest is then an out-of-domain application (see also Gildea, 2001).

In this paper, we present an evaluation of out-of-domain dependency parsing in the context of an interdisciplinary DH project. We evaluate the performance of six off-the-shelf dependency parsers trained on the *Hamburg Dependency Treebank* (HDT, Foth et al., 2014), a large German resource based on newscast texts by *heise online* with a focus on IT topics. Our test data consist of excerpts from one historic and one contemporary German novel as well as one contemporary German academic text. We aim to estimate the degree of error potentially introduced to our analyses by the out-of-domain application.

In addition to the overall evaluation, we focus on some dependency labels most relevant to further analysis for research questions in the humanities. For instance, our partners in literary studies want to learn about literary characters, i. e., the people acting in a novel. Therefore, explicit attributions are especially relevant, hence the quality of dependency labels relating character references to full verbs and also to non-verbal predicates (such as predicative nouns and adjectives) is of core interest.

Our results show good overall performance for most parsers with Mate (Bohnet, 2010) being the most successful one. However, performance on specific labels of interest to humanist scholars is rather poor.

2 Related Work

The first CoNLL multilingual shared task on dependency parsing was set up as an in-domain challenge and included German as one of the test languages (Buchholz and Marsi, 2006). In contrast to our work, the German training data was not based on a manually annotated dependency treebank but on a conversion of the Tiger treebank (Brants et al., 2004). The maximum spanning tree (MST) Parser (McDonald et al., 2006) scored best overall and also on the German test data, followed by an integer-linear programming (ILP) parser (Riedel et al., 2006) and the shift-reduce Malt parser (Nivre et al., 2006). We acknowledge this by including parsers in our evaluation that employ these three parsing paradigms, Malt parser itself being one of them. Furthermore, we rely on the following evaluation

Proceedings of the 14th Conference on Natural Language Processing (KONVENS 2018)
Vienna, Austria – September 19-21, 2018

metrics that are used in the shared task: attachment accuracies (UAS, LAS) and exact match.

Works on out-of-domain dependency parsing report mediocre results when parsers are applied to text types other than their training data. Foster et al. (2011) train Malt ('stackeager' algorithm) on Wall Street Journal texts and evaluate it on Twitter data. They find a "drastic" a drop in performance, which they reduce by domain adaptation.

Bohnet et al. (2012) taking part in the SANCL 2012 shared task on parsing of web texts (Petrov and McDonald, 2012) evaluate Mate (Bohnet, 2010) as a baseline system. They also train on Wall Street Journal texts but use different genres of web data as their testing domain. They report an average drop for LAS of 10.65 percentage points (from 90.54 % to 79.89 %) for baseline Mate.

For German, Ott and Ziai (2010) use Malt ('2-planar') to evaluate parser performance on a small data set of learner language. To this end, they train Malt on German newspaper texts. In particular, they employ a conversion (Versley, 2005) from the TüBa-D/Z treebank to the annotation scheme of the *Hamburg Dependency Treebank* (HDT), the latter being the very same treebank that we use in our evaluation. The LAS they achieve on learner texts is about four percentage points worse than the results on non-learner data. In their label-specific evaluation determiners, subjects, objects and auxiliaries score best. The detection of non-verbal predicates as well as the distinction between prepositional phrases as objects or adverbials is more difficult.

Foth et al. (2014) evaluate three statistical parsers on the original HDT: Malt ('2-planar'), Mate[1] and TurboParser (Martins et al., 2013), all three of which are also part of our study. They perform an in-domain evaluation and test for the impact of the training set size among others (using splits of 10, 100, 1,000, 10,000, 50,000 and 100,000 sentences respectively for training, and 1999 sentences for testing)[2]. Their general conclusion is that the more training data available the better the results. Overall, TurboParser (LAS of 93.57 %) and Mate (93.93 %) perform best while Malt ranks third (85.56 %). The algorithms of the first two benefit more from larger amounts of training data than Malt's shift-reduce paradigm. Since we train on the full 100,000 sentences training set of HDT, too, we expect Mate and TurboParser to have an advantage over Malt.

Sohl and Zinsmeister (2018) provide a small pilot study on out-of-domain dependency parsing by applying Malt, Mate (both trained on HDT) and JWCDG to textbook texts. They also report best results for Mate.

To our knowledge, our current work is the first in-depth evaluation of German dependency parsing in a digital humanities context that covers a range of area-specific text types and places particular emphasis on syntactic relationships considered most important for hermeneutic text interpretation.

3 Parsers

We experimented with six commonly-used, freely-available dependency parsers, which cover a certain variety of paradigms. Four of them are machine-learned, one is rule-based, and one has a hybrid architecture:

- **Malt**[3] (Nivre, 2003) parses under the paradigm of *shift-reduce* parsing: input token sequences are consumed linearly and along the way dependency relations are established. At each step, the next parser action is chosen based on the current parsing history.
- **Mate**[4] (Bohnet, 2010) treats dependency parsing as the construction of a maximum spanning tree, extending an algorithm by Carreras (2007).
- **RBGParser**[5] (Lei et al., 2014) employs tensor algebra to represent features. Parsing is done by sampling the space of possible syntax trees.
- **TurboParser**[6] (Martins et al., 2013) searches for a parse tree as a solution to an *integer linear programming* (ILP) problem.
- **JWCDG**[7] (Beuck et al., 2013) uses a grammar consisting of weighted hand-written constraints on morpho-syntactic and structural relationships which has been developed on the basis of the *Hamburg Dependency Treebank*.
- **ParZu**[8] (Sennrich et al., 2009; Sennrich et al., 2013) is a hybrid parser combining a hand-

[1] Mate is dubbed 'BohnetParser' in their paper, term suggested by the CLARIN-D project.

[2] We are only looking at their results for treebank part A, which contains manually corrected annotations.

[3] http://www.maltparser.org/

[4] https://code.google.com/archive/p/mate-tools/

[5] https://github.com/taolei87/RBGParser

[6] http://www.cs.cmu.edu/~ark/TurboParser/

[7] The CDG Team (1997–2015): https://gitlab.com/nats/jwcdg

[8] https://github.com/rsennrich/ParZu

written rule system with a probabilistic disambiguation component.

Apart from JWCDG and ParZu[9], whose grammars are hand-written, all parsers were trained on the first 100,000 sentences of the *Hamburg Dependency Treebank*, the same training set as used by Foth et al. (2014). The training data comprise 1,831,647 tokens (incl. punctuation) and the dependency annotation follows Foth (2006)[10]. We need training data following that annotation scheme so that the trained parsers yield parses comparable with those produced by JWCDG and ParZu, and the HDT is the largest available resource following that scheme. Unless noted otherwise, all parsers were trained with default settings, which means in particular that we did not use MaltOptimizer to fine-tune Malt's parsing performance.

Malt offers several algorithmic variants, differing inter alia in the predictor used to choose the next parser action. We evaluated all nine options, but will report only on the three of them that performed best on our data: 'stackeager', 'covnonproj', and 'stacklazy'.

The RBGParser and TurboParser support training models of different order. First-order models only involve information about single potential dependency edges (i. e. nodes and their immediate parent nodes), whereas higher-order models also consider features regarding more than a single potential dependency edge (second-order models, for example, take siblings and grandparents into account). We trained one model for each of the three predefined configurations 'basic', 'standard', and 'full', which, in this order, increasingly make use of higher-order features.

4 Test Data

The test data we use reflect the variety of text types we encounter in working with our project partners in the digital humanities project. They comprise extracts from the following texts:

- Modern literature (Lit2009): novel *Corpus Delicti: Ein Prozess* by German author Juli Zeh, published in Frankfurt/Main in 2009.
- Non-contemporary literature (Lit1850): *Eine Frauenfahrt um die Welt* ('A woman's journey around the world') by Austrian author Ida Pfeiffer (1850).[11]
- Modern academic writing (Aca2009): *Stand, Möglichkeiten und Grenzen der Telemedizin in Deutschland* ('Telemedicine in Germany: status, chances and limits') by Rüdiger Klar and Ernst Pelikan, published in Bundesgesundheitsblatt ('Federal Health Gazette') in 2009.
- Newscast (HDTtest): the last 1,999 sentences (i. e. all sentences not used for training) from part A of the *Hamburg Dependency Treebank* (Foth et al., 2014), i. e. in-domain data very similar to the training data.

The out-of-domain test texts comprise between 1,500 and 1,800 tokens.[12] Table 1 shows the great variation in sentence length between the text types: While the sentences in the contemporary literary text (Lit2009) are shorter on average, the historic literary text (Lit2850) shows a similar distribution of sentence lengths as HDTtest. The academic text (Aca2009) has much longer sentences and also a considerable amount of intratextual variation. Consequently, we expect the best performance on the contemporary literary text (Lit2009).

The test data were annotated manually following the annotation scheme of the HDT (Foth, 2006). The annotation was done by three annotators who had received some prior training on HDT data. They annotated independently section by section and met in between to discuss and adjudicate mismatches. See table 2 for the inter-annotator agreements (Fleiss, 1971) as measured on the annotations before discussion. Note that these numbers also include typos and errors due to lack of concentration, so they cannot be considered an upper bound for parser performance. The test data are available at https://doi.org/10.5281/zenodo.1324079.

While JWCDG is able to run directly on tokenized input (because it obtains the morphological information from its internal full-form dictionary and calls a tagger as part of the general disambiguation procedure), the other parsers require prior part-of-speech (POS) and morphological feature tagging as well as lemmatization. We experiment with two such inputs: manually annotated 'gold' POS tags and morphological features,

[9] We did not retrain ParZu's statistical component.

[10] See Foth et al. (2014) for an English summary.

[11] Full text available at Deutsches Textarchiv: http://www.deutschestextarchiv.de/pfeiffer_frauenfahrt01_1850/6.

[12] We have been hinted in the review process that this might be too little for reliable generalizations. Yet the data can show tendencies, although for more detailed analyses more data would be necessary.

text	tokens	sentences	mean sentence length	median sentence length	standard deviation	type-token ratio
Lit2009	1,518	114	13.32	12.0	8.66	0.46
Lit1850	1,647	82	20.10	18.5	11.56	0.46
Aca2009	1,790	75	24.00	22.0	18.42	0.44
HDTtest	40,975	1999	20.50	19.0	13.38	0.45

Table 1: Descriptive measures for the test texts (tokens including punctuation), TTR for HDTtest was calculated on the first 100 sentences (2001 tokens)

text	**head**	**deprel**	**head+deprel**
Lit2009	0.930	0.931	0.887
Lit1850	0.938	0.917	0.883
Aca2009	0.923	0.946	0.891
mean	0.931	0.931	0.887

Table 2: Inter-annotator agreement for the manual annotation of three annotators (Fleiss' κ)

and POS tags and morphological features computed by the ensemble approach in Adelmann et al. (2018). For lemmata, in both cases we use the output by JWCDG to be consistent with our training data.[13] ParZu can either be called with raw text, a raw tokenization, a POS-tagged token sequence (without morphology), or fully preprocessed data; however, the required preprocessing generates richer and more detailed information than available in our morphological annotation, so we called ParZu with our tokenization and POS-tags and had it determine morphological information itself.

While one might argue that in practice gold-standard POS and morphology inputs are a highly unrealistic setting given that manually annotating them is nearly as time-consuming as manual dependency annotation, evaluation metrics for parses based on gold-standard inputs reflect only parser performance itself and can help distinguish errors inherent to parsers from errors that stem from imperfect prior processing.

5 Results

We examine our parsing results with respect to three dimensions: parser performance, differences between text types, and out-of-vocabulary phenomena.

Furthermore, we are interested in specific phenomena potentially important for further analyses in a digital humanities setting. Overall performance measures such as attachment accuracies capture many phenomena that are frequent, but usually uninteresting for text interpretation. For instance, articles are ubiquitous in German texts and therefore contribute considerably to the overall accuracy. As their correct attachment does not pose a major problem for modern parsers, they are beneficial to statistical performance measures; yet, articles are of little interest beyond purely linguistic concerns.

For the overall evaluation, we record the following common performance measures:

- **Exact match**: the fraction of sentences parsed completely correctly.
- **UAS** and **LAS**: the unlabeled and labeled attachment score (accuracy), respectively. They are the overall fraction of tokens having been assigned the correct head (UAS), or head as well as dependency relation (LAS).
- **Deprel accuracy**: the fraction of tokens having been assigned the correct dependency relation to their respective head (regardless of whether the head itself is correct).

We report the attachment accuracies as whole percentage numbers as our test data is too small for decimal places to be meaningful. All measures we report exclude punctuation[14] for similar reasons as given above regarding articles.

For JWCDG, there is only one result per text; for the other parsers, there are two (one based on gold part-of-speech and morphology, one based on automatically tagged part-of-speech and morphology). Hence there is one table for JWCDG (5) and two tables (3, 4) for the other parsers. Recall that

[13] HDT's lemma annotations are extracted from the parser's full-form dictionary or in case of out-of-vocabulary words generated automatically by a simple stemmer.

[14] The proportion of punctuation in our test texts is as follows: Lit2009: 16.7 %, Lit1850: 13.0 %, Aca2009: 11.6 %, HDTtest: 12.4 %.

ParZu received only part-of-speech tags from our annotation, but no morphological information (as explained in section 4). The tables reside in the appendix.

5.1 Parser Comparison

If one were to declare a winner, chances are it would be Mate. As far as the overall attachment accuracies (both labeled and unlabeled) are concerned, Mate is usually among the highest-rated parsers, only in one case (Lit2009, gold morphology) being beaten by RBGParser's 'full' model and in two other cases (Lit2009 and Aca2009, computed morphology) by Malt's 'covnonproj'. However, Mate's lead over the second best parser (RBG or Malt usually) is small under all conditions (especially when labeled accuracies are considered). For none of the text types do we observe a deviation of more than ten percentage points between parsers.

When considering exact matches, the picture is not that clear. Exact match is a sensitive measure, requiring the whole syntax tree to be correct, and hence exact match accuracies are generally low. Mate still notably surpasses the majority of other parsers (e. g. 53 % for Lit2009 with gold morphology, the other parsers giving 30 % to 45 %), but there are more cases of Mate being outperformed. Systematically, ParZu offers better exact match accuracies for Lit1850 and Aca2009 (for both gold and computed morphology), being by far (3 to 6 percentage points difference) best for this metric in those cases.

A comparison between the three Malt algorithms included in this paper identifies 'covnonproj' as the best, giving higher attachment and dependency relation accuracies than the two others throughout the tables. Yet for HDTtest and Lit2009, both give higher exact match accuracies. In general, 'stacklazy' performs slightly better than 'stackeager'.

The data also show that rule-based parsers are more robust against out-of-domain texts[15]. The attachment accuracy of the rule-based JWCDG decreases by at most three percentage points between the in-domain HDTtest and the other texts (in three cases, they even rise). This is a modest performance drop compared to the trained parsers where attachment accuracy decreases by up to 16 percentage points in some cases (TurboParser for Aca2009 with gold-morphology input), which confirms the results of previous work reported in section 2. ParZu is not entirely rule-based, but its performance drop is similar to that of JWCDG. However, the extent to which the attachment accuracy actually decreases varies greatly between parsers and texts, and a performance impact cannot always be observed. As for exact match though, a peculiarity of ParZu can be observed: While all other parsers (including JWCDG) show considerable fluctuations across texts, ParZu's values remain relatively constant (this is even clearer for the computed-morphology input than for the gold-morphology input). One consequence is the above-mentioned phenomenon that ParZu, although being average at best on the other texts, outperforms the other parsers (including JWCDG) on Lit1850 and Aca2009, where the latter ones fail to produce large amounts of exact matches.

5.2 Text Comparison

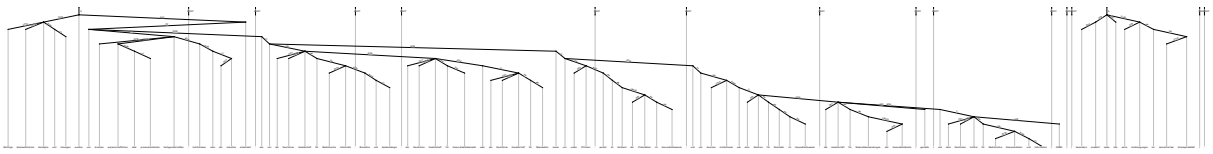
For both gold morphology and computed morphology as input, we see a notable performance drop for the 19th-century literary text (Lit1850) and the modern academic writing (Aca2009) compared with the other texts (exact match: up to 38 percentage points; UAS/LAS: up to 16 percentage points). For gold morphology, Aca2009 gives even worse results than Lit1850; for computed morphology, both score approximately equal. Both texts have some characteristics that deviate from the training data and can potentially explain this difference: The 19th-century text has some specific syntactic structures, e. g. relative pronouns rather uncommon in written language today:

(1) ein Priester, **welchem** zur Seite ein Chinese mit einer zwei Fuß hohen Laterne ging, …

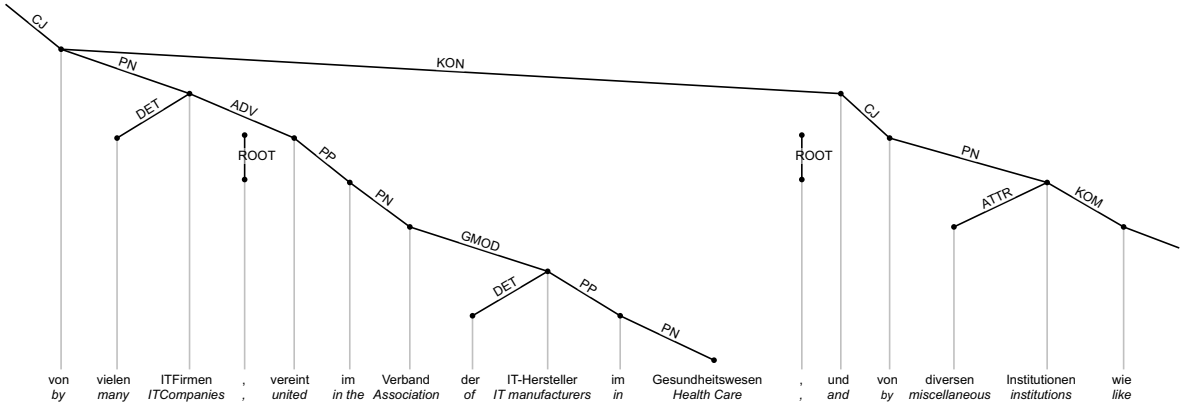
'a priest, at whose side walked a Chinese with a two-foot-high lantern, …'

The modern academic text exhibits a notable amount of complex nested prepositional and coordinated noun phrases (fig. 1), which are often inherently ambiguous and thus hard to attach correctly. Dependency relation accuracy did not vary much though, i. e. the parsers assigned relation labels almost equally well for all texts. Sometimes, dependency relation accuracies were even slightly higher for the 'problematic' texts discussed above.

[15] Even for non-trained parsers it makes sense to speak of in-domain data if they are designed for a particular data set. JWCDG's rules, for instance, were manually optimized for the HDT.



(a) The full gold-standard syntax tree of the sentence, bird's eye view. The sentence has 101 tokens.



(b) Excerpt from the syntax tree, with translation, worm's-eye view.

Figure 1: A highly complex example sentence with nested coordinated phrases from Aca2009.

Interestingly, some parsers performed better on the contemporary literature (Lit2009) than the in-domain HDT test data (HDTtest). Although this is hardly surprising given the (on average) shorter sentence length, it is still noteworthy. With gold POS and morphology, exact match increases by 1 to 3 percentage points for all three Malt algorithms. With computed POS and morphology, we see an exact match increase for all parsers (up to 19 percentage points, for RBGParser's 'standard' and 'full' models) as well as an attachment accuracy increase, again for all parsers (UAS: up to 4 percentage points, for RBGParser's and TurboParser's 'basic' models as well as ParZu; LAS: up to 3 percentage points, for ParZu). To some extent this phenomenon can also be observed for the rule-based JWCDG, where Lit2009's UAS is five percentage points higher than HDTtest's. This can again be explained by text characteristics: Lit2009 contains numerous elliptic sentences, as reflected in the low sentence length (table 1). In example (2) one short complete sentence is followed by two elliptical expressions:

(2) Das verstehe ich. Aber trotzdem. Das eigene Kind!

'I understand that. But still. One's own child!'

Elliptical sentences are rare in the newscast texts of HDT, but easier to assign a correct syntax tree. This is consistent with the observation of a higher increase in the exact match score than in the attachment accuracy, which measures local assignment decisions and is therefore independent of sentence length and complexity.

5.3 Out-of-Vocabulary Phenomena

It is normal for test data in language processing to contain a number of out-of-vocabulary (OOV) tokens, i. e. tokens whose word form or lemma (depending on the task to solve) never occurred in the training data. One has to expect this situation to be even more common when the training data (newscast, in our case) are from a substantially different domain than the test data, as it is often the case in digital humanities projects.

We count tokens as OOV if their lemma[16] does not occur in the HDT training data, and observe an interesting phenomenon. When computing attachment accuracies[17] only for those tokens as basic population, one would expect these values to decrease since lexical information as a feature is rendered useless. However, in our case, we unexpectedly observe *increases* in several cases, especially where gold POS and morphology input is

[16] Recall that all lemma information we use is from JWCDG's full-form lexicon.

[17] Note that we cannot compute an 'exact match' measure here as we do not have whole sentences.

used: While dependency relation accuracy usually decreases, attachment scores for HDTtest increase for all parsers (most notably for Malt's 'stack-eager': UAS from 88 % to 93 % and LAS from 87 % to 90 %). For the out-of-domain texts, the situation is more mixed, but dependency relation accuracy usually deteriorates here, too. The parsers whose UAS and LAS increase differ from text to text, but among those whose UAS deteriorates we always find RBGParser and among those whose UAS does not deteriorate we always find TurboParser. For example, when looking at Lit2009, TurboParser's 'basic' and 'standard' models are the only cases without decreasing UAS, and when looking at Aca2009, RBGParser is the only parser without increasing UAS (cf. table 7 in the appendix).

Where computed POS and morphology input is used, attachment accuracy increases are less common (see table 8 in the appendix), but considerable amounts can be observed for TurboParser (LAS increase from 76 % to 84 % for 'basic' Aca2009). These facts suggest to us that

- RBGParser is not good at generalizing to unknown words, while TurboParser is;
- at least for the models used by the parsers we tested, lexical information (i. e. the lemma) has considerably less impact than other features like POS and morphology;[18]
- lexical information seems to be even able to impair parsing performance, at least when POS and morphology quality is high.

One should note, though, that possible attachments of a token are never assessed with respect to only that token, but to at least the potential head tokens as well. The case that both a token and its (correct) head are out-of-vocabulary occurs almost never in our test data (and is presumably rare in other texts, too). The morphological features of a token, the lexical and morphological features of a potential head, and possibly, in case of higher-order features, even broader syntactic structures can provide a parser with enough information needed to compensate for missing lexical information of a single token and thus to be robust against out-of-vocabulary words; parsers differ in how well they use that information.

OOV rates range from 1.86 % (HDTtest, 764 tokens OOV) to 5.46 % (Lit1850, 75 tokens OOV). A closer look at the OOV tokens in the individual texts reveals no notable differences in the dependency labels the OOV tokens are labeled with, but their POS tags show different distributions, which can partially be related to text characteristics. While `NN` (normal noun), `NE` (named entity) and other tags of open word classes constitute the vast majority in all texts, there are more `NE` OOV tokens in HDTtest, where also `FM` (foreign-language material) OOV tokens are prominent (the only other test text with `FM` OOV tokens, and much less of them, is Aca2009). This is not surprising as there are many IT-related technical terms (predominantly in English) in the HDT. The amount of OOV verbs is surprisingly low in Lit2009 and Aca2009. In HDTtest there is a notable amount of OOV infinitives and past participles, while in Lit1850 the OOV verbs mainly comprise finite forms. This might be related to some kind of text characteristic, but we do not know which. In Lit1850, only 3 out of 18 OOV finite verb forms are obsolete spellings and two others are irregular inflections not used anymore today (*bekömmt*, modern *bekommt* 'receives', and *gebeut*, modern *gebietet*, 'demand', 'command', 'require').

5.4 Label-Specific Evaluation

Our special focus is on those labels that are important for further analysis in a digital humanities project. For instance, literary scholars are interested in the automatic generation of character profiles[19] for a novel. This can be achieved in various ways, for example by looking at character traits explicitly attributed to a character or by listing all the full verbs that cooccur with a character in subject position. Consequently, it is important to us that the verbal structure and its main arguments are identified correctly. This section will consider the results for the out-of-domain texts and the automatically generated ensemble part-of-speech tags and morphology as input only, as this is a realistic setting for a digital humanities project. We report F1 scores as the harmonic mean of precision and recall of individual labels.

Explicit attribution of character traits can be found in non-verbal predicates such as in (3), in which the property of being a nice boy is syntactically encoded in a nominal predicate.

[18] Recall that for both gold-standard and computed POS and morphology input we used the same lemmatization.

[19] Here, again, we refer to the literary concept of a character as a person in a text.

(3) Rosentreter ist ein netter Junge
'Rosentreter is a nice boy'

However, the results for non-verbal predicates (label PRED) are rather mediocre (see also Ott and Ziai, 2010). Non-verbal predicates are frequently confused with subjects as they are also in nominative case and can be fronted in German. Additionally, confusions with adverbials are common as this distinction is based on the semantic properties of the verb. Malt with the 'covnonproj' model achieves the best scores for this task (F1 scores between 0.71 and 0.79), while for TurboParser, RBG and Mate all F1 scores are 0.32 or lower. Note that due to the small test data sets of 75–114 sentences, the absolute numbers of non-verbal predicates in the three out-of-domain texts are 14, 14, and 23 only, so we can only report tentative results. In contrast to identifying the correct label, choosing the correct attachment point for non-verbal predicates is not a problem.

Explicit characterization can also take the form of appositions, e. g. *die sorgliche Hausfrau, Mad. Behn* ('the caring house wife, Mad. Behn'). The same label (APP) is used to connect parts of proper names[20]. Here, too, Malt ('covnonproj') scores best (F1: 0.78–0.80). Again, the absolute number of instances in the test texts is low (9, 9, 29).

A more indirect way of characterization can be obtained by looking at the verbal structures that a specific character occurs in. The relevant labels are much more frequent in the texts than the non-verbal predicates and the parsing results are better. The core verbal structure comprises the correct detection of the root of the sentence with the label S and, in the case of subordinate clauses, REL for relative clauses and NEB for adverbial clauses[21].

For the label S, the F1 scores are high, between 0.66 and 0.93. In contrast to the other labels, TurboParser and Mate achieve the best scores (F1 between 0.89 and 0.93 for both). The corresponding attachment scores are even higher than the F1 scores in almost all cases.

The results are slightly worse for subordinate clauses, but here too Mate achieves by far the best results (between 0.86 and 0.96). The other parsers drop as low as 0.55 (JWCDG on the 19th-century text Lit1950). While the F1 scores are somewhat better for relative clauses, the attachment scores are poor. Where the label REL has been attributed correctly, the attachment scores are between 0.43 and 0.92, again with Mate in first place, followed by ParZu and JWCDG, all on the 19th-century text. The scores are clearly worst for the academic text, which is easily explained by the many complex noun phrases with several possibilities for attachment (cf. section 5.2).

For the detection of complex verbs (such as compound tenses), the label AUX is important. The scores for this core label are very good, no parser dropping below 0.80. TurboParser, Mate, Malt (covnonproj), ParZu and RBGParser perform similarly. The clearer difference is in text type: The 19th-century text (Lit1850) does not achieve any better score than 0.86, possibly due to cases like example (4), which features a complex verb phrase and outdated spelling of the main verb *residieren* 'reside'.

(4) …, wo noch vor wenig Tagen solch ein Ungeheuer residirt haben sollte.
'…, where only a few days ago such a monster should have resided.'

In addition to the verbal complex itself, let us finally have a look at the most important arguments: subject (SUBJ) and direct object (OBJA). The results for subject detection are slightly worse than the results of the evaluation across all labels, with ParZu performing best this time (0.87 to 0.93). The scores for direct objects are lower. In particular, all three Malt models achieve rather low F1 scores, which is mostly due to a low precision. The best parser is again ParZu (0.77 to 0.86).

In summary, we have seen that the parsers show individual strengths that should be taken into consideration when using a parser for a specific purpose.

6 Conclusion

We have demonstrated that Mate achieves the best overall result on out-of-domain texts in our study. As expected, the performance of the parsers depends greatly on the syntactic complexity (measured in sentence length) of the text. Therefore, the contemporary literary (Lit2009) text with short sentences sometimes scores even better than the in-domain text. The 19th-century text (Lit1850)

[20] The label is also used for phrases like *Raum 20/09* ('room 20/09') or *den Buchstaben F* ('the letter F'), see Foth (2006).

[21] Subject and object clauses (SUBJC and OBJC) fall into the same category, but occur too rarely in our test data for a meaningful evaluation.

and the academic text (Aca2009) have longer sentences and achieve much lower scores.

When looking at labels that are of special interest to a scholar in the humanities, we are left with partly low scores, for example, for the detection of non-verbal predicates and appositions. For these tasks, Malt with the model 'covnonproj' yields considerably better (though still not great) results. Detection of the main verbal structure and its core arguments works much better for all parsers. As far as the analysis in our DH project is concerned, we conclude that even though non-verbal predicates promise direct access to character traits, their automatic detection and attribution is unreliable and should be double-checked; we also have to expect to miss a considerable amount of them. It might be worth using the Malt model for this specific task, or employing a parser ensemble (e. g. Sagae and Tsujii, 2007). Surdeanu and Manning (2010) show that even simple majority-vote ensembles are usually sufficient to improve parsing quality. How much of an improvement can be achieved with respect to non-verbal predicates has yet to be investigated.

Acknowledgements

This work has been funded by the 'Landesforschungsförderung Hamburg' in the context of the *hermA* project (LFF-FV 35). We would like to thank the reviewers for their thorough comments, Lea Röseler and Sarah Jablotschkin for their invaluable annotation effort and Piklu Gupta for improving our English. All remaining errors are ours.

References

Benedikt Adelmann, Melanie Andresen, Wolfgang Menzel, and Heike Zinsmeister. 2018. Evaluating Part-of-Speech and Morphological Tagging for Humanities' Interpretation. In Andrew Frank, Christine Ivanovic, Francesco Mambrini, Marco Passarotti, and Caroline Sporleder, editors, *Proceedings of the Second Workshop on Corpus-Based Research in the Humanities (CRH-2)*, volume 1 of *Gerastree proceedings*, pages 5–14, Vienna, Austria.

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*, pages 186–206. IOS press.

Bernd Bohnet, Richárd Farkas, and Özlem Çetinoğlu. 2012. SANCL 2012 shared task: The IMS system description. In *Workshop on the Syntactic Analysis of Non-Canonical Language (SANCL 2012)*. See https://sites.google.com/site/sancl2012/home/programme.

Bernd Bohnet. 2010. Very high accuracy and fast dependency parsing is not a contradiction. In *Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010)*, pages 89–97, Beijing, China.

Sabine Brants, Stefanie Dipper, Peter Eisenberg, Silvia Hansen-Schirra, Esther König, Wolfgang Lezius, Christian Rohrer, George Smith, and Hans Uszkoreit. 2004. TIGER: Linguistic interpretation of a German corpus. *Research on language and computation*, 2(4):597–620.

Sabine Buchholz and Erwin Marsi. 2006. CoNLL-X shared task on multilingual dependency parsing. In *Proceedings of the Tenth Conference on Computational Natural Language Learning (CoNLL-X)*, pages 149–164, New York, NY, USA.

Xavier Carreras. 2007. Experiments with a higher-order projective dependency parser. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 957–961, Prague, Czech Republic.

Joseph L. Fleiss. 1971. Measuring nominal scale agreement among many raters. *Psychological Bulletin*, 76(5):378–382.

Jennifer Foster, Özlem Çetinoglu, Joachim Wagner, Joseph Le Roux, Stephen Hogan, Joakim Nivre, Deirdre Hogan, and Josef van Genabith. 2011. #hardtoparse: POS Tagging and Parsing the Twitterverse. In *AAAI 2011 Workshop On Analyzing Microtext*, pages 20–25, San Francisco, CA, USA.

Kilian Foth, Arne Köhn, Niels Beuck, and Wolfgang Menzel. 2014. Because Size Does Matter: The Hamburg Dependency Treebank. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC 2014)*, pages 2326–2333, Reykjavik, Iceland.

Kilian Foth. 2006. Eine umfassende Constraint-Dependenz-Grammatik des Deutschen. Available at http://edoc.sub.uni-hamburg.de/informatik/volltexte/2014/204/.

Daniel Gildea. 2001. Corpus Variation and Parser Performance. In *Proceedings of the 2001 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Tao Lei, Yu Xin, Yuan Zhang, Regina Barzilay, and Tommi Jaakkola. 2014. Low-rank tensors for scoring dependency structures. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1381–1391, Baltimore, MD, USA.

André 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.

Ryan McDonald, Kevin Lerman, and Fernando Pereira. 2006. Multilingual dependency analysis with a two-stage discriminative parser. In *Proceedings of the Tenth Conference on Computational Natural Language Learning (CoNLL-X)*, pages 216–220, New York, NY, USA.

Joakim Nivre, Johan Hall, Jens Nilsson, Gülşen Eryiğit, and Svetoslav Marinov. 2006. Labeled pseudo-projective dependency parsing with support vector machines. In *Proceedings of the Tenth Conference on Computational Natural Language Learning (CoNLL-X)*, pages 221–225, New York, NY, USA.

Joakim Nivre. 2003. An efficient algorithm for projective dependency parsing. In *Proceedings of the 8th International Workshop on Parsing Technologies (IWPT)*, pages 149–160, Nancy, France.

Niels Ott and Ramon Ziai. 2010. Evaluating Dependency Parsing Performance on German Learner Language. In *Proceedings of the Ninth International Workshop on Treebanks and Linguistic Theories*, pages 175–186, Tartu, Estonia.

Slav Petrov and Ryan McDonald. 2012. Overview of the 2012 shared task on parsing the web. In *Workshop on the Syntactic Analysis of Non-Canonical Language (SANCL 2012)*. See https://sites.google.com/site/sancl2012/home/programme.

Sebastian Riedel, Ruket Cakici, and Ivan Meza-Ruiz. 2006. Multi-lingual dependency parsing with incremental integer linear programming. In *Proceedings of the Tenth Conference on Computational Natural Language Learning (CoNLL-X)*, pages 226–230, New York, NY, USA.

Kenji Sagae and Jun'ichi Tsujii. 2007. Dependency parsing and domain adaptation with LR models and parser ensembles. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 1044–1050, Prague, Czech Republic.

Rico Sennrich, Gerold Schneider, Martin Volk, and Martin Warin. 2009. A New Hybrid Dependency Parser for German. In Christian Chiarcos, Richard Eckart de Castilho, and Manfred Stede, editors, *Von der Form zur Bedeutung: Texte automatisch verarbeiten / From Form to Meaning: Processing Texts Automatically. Proceedings of the Biennial GSCL Conference 2009*, pages 115–124, Tübingen. Narr.

Rico Sennrich, Martin Volk, and Gerold Schneider. 2013. Exploiting Synergies Between Open Resources for German Dependency Parsing, POS-tagging, and Morphological Analysis. In *Proceedings of the International Conference Recent Advances in Natural Language Processing (RANLP 2013)*, pages 601–609, Hissar, Bulgaria.

Jessica Sohl and Heike Zinsmeister. 2018. Exploring ensemble dependency parsing to reduce manual annotation workload. In Georg Rehm and Thierry Declerck, editors, *Language Technologies for the Challenges of the Digital Age*, pages 40–47, Cham. Springer International Publishing.

Mihai Surdeanu and Christopher D. Manning. 2010. Ensemble models for dependency parsing: cheap and good? In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 649–652, Los Angeles, CA, USA.

Yannick Versley. 2005. Parser evaluation across text types. In *Proceedings of the Fourth Workshop on Treebanks and Linguistic Theories (TLT-2005)*, Barcelona, Spain.

Appendix

Text	Parser	model type (resp. order)	exact match	UAS	LAS	deprel accuracy
HDTtest	Malt	stackeager	0.38	88 %	87 %	0.93
		covnonproj	0.35	92 %	90 %	0.94
		stacklazy	0.38	90 %	89 %	0.94
	Mate	–	**0.56**	**96 %**	**96 %**	**0.98**
	RBG	basic	0.49	95 %	94 %	**0.98**
		standard	0.53	**96 %**	95 %	**0.98**
		full	0.54	**96 %**	95 %	**0.98**
	Turbo	basic	0.44	94 %	92 %	0.97
		standard	0.47	95 %	93 %	0.97
		full	0.47	95 %	94 %	0.97
	ParZu	–	0.31	90 %	87 %	0.93
Lit2009	Malt	stackeager	0.41	85 %	82 %	0.89
		covnonproj	0.36	90 %	85 %	0.90
		stacklazy	0.41	88 %	84 %	0.90
	Mate	–	**0.53**	91 %	**89 %**	**0.93**
	RBG	basic	0.39	89 %	86 %	0.90
		standard	0.44	91 %	88 %	0.91
		full	0.45	**92 %**	88 %	0.92
	Turbo	basic	0.30	85 %	80 %	0.87
		standard	0.35	88 %	84 %	0.90
		full	0.36	89 %	84 %	0.90
	ParZu	–	0.40	88 %	85 %	0.90
Lit1850	Malt	stackeager	0.16	82 %	78 %	0.87
		covnonproj	0.20	88 %	82 %	0.89
		stacklazy	0.17	86 %	80 %	0.87
	Mate	–	0.29	**91 %**	**87 %**	**0.93**
	RBG	basic	0.13	86 %	82 %	0.89
		standard	0.15	89 %	84 %	0.90
		full	0.20	90 %	85 %	0.91
	Turbo	basic	0.15	82 %	77 %	0.85
		standard	0.21	87 %	82 %	0.88
		full	0.22	88 %	82 %	0.88
	ParZu	–	**0.33**	89 %	85 %	0.90
Aca2009	Malt	stackeager	0.17	84 %	81 %	0.90
		covnonproj	0.17	88 %	**85 %**	0.92
		stacklazy	0.17	86 %	83 %	0.91
	Mate	–	0.18	**89 %**	**85 %**	**0.93**
	RBG	basic	0.11	83 %	79 %	0.89
		standard	0.18	86 %	83 %	0.91
		full	0.17	88 %	84 %	0.91
	Turbo	basic	0.11	80 %	76 %	0.86
		standard	0.11	83 %	79 %	0.88
		full	0.11	84 %	80 %	0.89
	ParZu	–	**0.22**	87 %	84 %	0.92

Table 3: Performance measures for the parsers (excluding JWCDG) on our test texts, with gold-standard part-of-speech and (except for ParZu) morphological features as input.

Text	Parser	model type (resp. order)	exact match	UAS	LAS	deprel accuracy
HDTtest	Malt	stackeager	**0.26**	84 %	80 %	0.87
		covnonproj	0.23	**88 %**	**84 %**	**0.89**
		stacklazy	**0.26**	86 %	82 %	0.88
	Mate	–	0.24	**88 %**	**84 %**	**0.89**
	RBG	basic	0.15	82 %	78 %	0.84
		standard	0.18	86 %	81 %	0.86
		full	0.19	87 %	81 %	0.87
	Turbo	basic	0.13	81 %	76 %	0.82
		standard	0.20	86 %	80 %	0.85
		full	0.16	86 %	80 %	0.85
	ParZu	–	0.21	85 %	81 %	0.87
Lit2009	Malt	stackeager	0.38	87 %	82 %	**0.89**
		covnonproj	0.33	**90 %**	**84 %**	0.88
		stacklazy	0.37	88 %	83 %	0.88
	Mate	–	**0.41**	89 %	**84 %**	**0.89**
	RBG	basic	0.31	86 %	80 %	0.85
		standard	0.37	89 %	83 %	0.87
		full	0.38	89 %	83 %	0.87
	Turbo	basic	0.27	85 %	78 %	0.83
		standard	0.29	88 %	80 %	0.86
		full	0.31	89 %	81 %	0.86
	ParZu	–	0.35	89 %	**84 %**	**0.89**
Lit1850	Malt	stackeager	0.15	78 %	73 %	0.84
		covnonproj	0.16	84 %	79 %	0.86
		stacklazy	0.17	82 %	76 %	0.84
	Mate	–	0.24	**87 %**	**82 %**	**0.88**
	RBG	basic	0.11	82 %	76 %	0.84
		standard	0.11	85 %	79 %	0.85
		full	0.15	86 %	80 %	0.85
	Turbo	basic	0.09	80 %	72 %	0.81
		standard	0.12	84 %	76 %	0.83
		full	0.11	84 %	76 %	0.83
	ParZu	–	**0.27**	85 %	80 %	0.85
Aca2009	Malt	stackeager	0.17	82 %	79 %	0.89
		covnonproj	0.15	**85 %**	**82 %**	**0.90**
		stacklazy	0.15	**85 %**	81 %	**0.90**
	Mate	–	0.15	**85 %**	81 %	0.89
	RBG	basic	0.09	80 %	77 %	0.86
		standard	0.15	84 %	80 %	0.88
		full	0.17	84 %	80 %	0.88
	Turbo	basic	0.09	77 %	73 %	0.84
		standard	0.11	81 %	76 %	0.86
		full	0.11	82 %	77 %	0.86
	ParZu	–	**0.23**	84 %	81 %	0.89

Table 4: Performance measures for the parsers (excluding JWCDG) on our test texts, with automatically annotated part-of-speech and (except for ParZu) morphological features as input.

Text	exact match	UAS	LAS	deprel accuracy
HDTtest	0.24	82 %	79 %	0.87
Lit2009	0.41	87 %	82 %	0.87
Lit1850	0.18	82 %	77 %	0.85
Aca2009	0.08	79 %	76 %	0.88

Table 5: Performance measures for JWCDG on our test texts.

Text	Parser	APP	AUX	NEB	OBJA	OBJC	OBJD	OBJI	PRED	REL	S	SUBJ
HDTtest	Mate	0.87	0.93	0.71	0.74	0.67	0.51	0.70	**0.44**	0.74	0.87	0.84
	ParZu	0.85	0.91	0.60	0.83	0.67	0.67	0.65	0.72	0.76	0.69	0.87
	JWCDG	0.75	0.89	0.70	0.76	0.51	0.56	0.71	0.62	0.77	0.80	0.83
	Malt, covnonproj	0.88	0.93	0.69	0.76	0.62	**0.28**	0.67	0.75	0.75	0.88	0.82
	Malt, stackeager	0.84	0.85	0.55	0.72	**0.46**	**0.33**	0.53	0.67	0.58	0.82	0.81
	Malt, stacklazy	0.86	0.88	0.62	0.77	0.52	**0.32**	0.56	0.77	0.61	0.79	0.83
	RBGParser, full	0.82	0.92	0.61	0.70	0.60	**0.44**	0.72	**0.31**	0.67	0.85	0.80
	TurboParser, full	0.81	0.92	0.52	0.68	0.63	**0.32**	0.64	**0.32**	0.62	0.89	0.76
Lit2009	Mate	0.60	0.94	0.96	0.82	0.86	0.52	0.88	**0.31**	0.77	0.91	0.88
	ParZu	0.67	0.95	0.86	0.86	0.57	0.80	0.93	0.67	0.88	0.80	0.93
	JWCDG	0.52	0.92	0.69	0.86	0.57	0.76	0.55	0.64	0.73	0.83	0.85
	Malt, covnonproj	0.80	0.94	0.89	0.75	0.57	**0.40**	0.86	0.79	0.77	0.83	0.87
	Malt, stackeager	0.69	0.85	0.86	0.77	**0.40**	**0.40**	0.75	0.73	0.71	0.90	0.86
	Malt, stacklazy	0.67	0.86	0.74	0.75	**0.29**	**0.40**	0.80	0.71	0.78	0.89	0.87
	RBGParser, full	0.58	0.94	0.73	0.75	0.75	**0.45**	0.93	**0.22**	0.64	0.87	0.86
	TurboParser, full	0.67	0.94	0.81	0.78	0.75	0.50	0.80	**0.17**	0.77	0.89	0.81
Lit1850	Mate	0.56	0.86	0.89	0.76	0.73	0.72	0.89	**0.32**	0.82	0.93	0.89
	ParZu	0.63	0.86	0.72	0.85	0.62	0.74	0.76	0.70	0.64	0.66	0.88
	JWCDG	**0.48**	0.82	0.55	0.78	**0.44**	0.61	0.78	0.60	0.86	0.83	0.87
	Malt, covnonproj	0.78	0.86	0.80	0.71	**0.40**	0.51	0.74	0.71	0.81	0.92	0.84
	Malt, stackeager	0.67	0.83	0.67	0.69	**0.22**	0.52	0.70	0.62	0.71	0.84	0.78
	Malt, stacklazy	0.57	0.80	0.67	0.69	**0.46**	**0.36**	0.74	0.70	0.72	0.84	0.83
	RBGParser, full	0.67	0.83	0.57	0.79	**0.36**	0.74	0.80	**0.19**	0.76	0.89	0.86
	TurboParser, full	**0.43**	0.82	0.58	0.71	0.57	0.55	0.73	**0.11**	0.67	0.93	0.82
Aca2009	Mate	0.76	0.95	0.86	0.65	0.80	0.50	0.67	**0.25**	0.86	0.89	0.80
	ParZu	0.79	0.89	0.64	0.77	0.67	**0.22**	0.82	0.77	0.80	0.73	0.87
	JWCDG	0.54	0.87	0.76	0.62	0.80	0.57	0.63	0.61	0.93	0.85	0.86
	Malt, covnonproj	0.78	0.96	0.67	0.72	0.80	**0.33**	0.75	0.76	0.88	0.91	0.77
	Malt, stackeager	0.68	0.86	0.60	0.51	0.80	**0.15**	0.57	0.67	0.78	0.88	0.75
	Malt, stacklazy	0.68	0.91	0.78	0.75	**0.33**	**0.33**	0.57	0.71	0.61	0.84	0.80
	RBGParser, full	0.75	0.94	0.84	0.60	0.67	**0.33**	0.67	**0.14**	0.88	0.86	0.76
	TurboParser, full	0.80	0.98	0.74	0.59	0.57	**0.36**	0.63	**0.11**	0.80	0.92	0.65

Table 6: Label-specific performances (F1 scores): Weak results with F1 score < 0.5 are marked in bold.

Text	Parser	model type (resp. order)	UAS		UAS OOV	LAS		LAS OOV	deprel		deprel OOV
HDTtest	Malt	stackeager	88 %	↗	93 %	87 %	↗	90 %	0.93	↘	0.92
		covnonproj	92 %	↗	94 %	90 %	→	90 %	0.94	↘	0.91
		stacklazy	90 %	↗	93 %	89 %	↗	90 %	0.94	↘	0.91
	Mate	–	96 %	↗	98 %	96 %	↗	98 %	0.98	→	0.98
	RBG	basic	95 %	↗	98 %	94 %	↗	96 %	0.98	↘	0.97
		standard	96 %	↗	98 %	95 %	↗	96 %	0.98	↘	0.97
		full	96 %	↗	98 %	95 %	↗	96 %	0.98	↘	0.97
	Turbo	basic	94 %	↗	97 %	92 %	↗	96 %	0.97	→	0.97
		standard	95 %	↗	97 %	93 %	↗	95 %	0.97	→	0.97
		full	95 %	↗	97 %	94 %	↗	96 %	0.97	→	0.97
Lit2009	Malt	stackeager	85 %	↘	83 %	82 %	↘	80 %	0.89	↘	0.84
		covnonproj	90 %	↘	88 %	85 %	↗	87 %	0.90	↘	0.89
		stacklazy	88 %	↘	83 %	84 %	↘	79 %	0.90	↘	0.84
	Mate	–	91 %	↘	89 %	89 %	↘	84 %	0.93	↘	0.87
	RBG	basic	89 %	↘	87 %	86 %	↘	81 %	0.90	↘	0.84
		standard	91 %	↘	87 %	88 %	↘	81 %	0.91	↘	0.84
		full	92 %	↘	89 %	88 %	↘	84 %	0.92	↘	0.87
	Turbo	basic	85 %	↘	84 %	80 %	↘	73 %	0.87	↘	0.80
		standard	88 %	→	88 %	84 %	↘	77 %	0.90	↘	0.84
		full	89 %	↘	84 %	84 %	↘	75 %	0.90	↘	0.81
Lit1850	Malt	stackeager	82 %	→	82 %	78 %	→	78 %	0.87	↘	0.81
		covnonproj	88 %	→	88 %	82 %	↗	83 %	0.89	↘	0.86
		stacklazy	86 %	↘	83 %	80 %	↘	78 %	0.87	↘	0.79
	Mate	–	91 %	↗	92 %	87 %	→	87 %	0.93	↘	0.89
	RBG	basic	86 %	↘	81 %	82 %	↘	73 %	0.89	↘	0.76
		standard	89 %	↘	87 %	84 %	↘	77 %	0.90	↘	0.79
		full	90 %	→	90 %	85 %	↘	80 %	0.91	↘	0.82
	Turbo	basic	82 %	↘	78 %	77 %	↘	67 %	0.85	↘	0.69
		standard	87 %	↗	88 %	82 %	↘	78 %	0.88	↘	0.81
		full	88 %	↗	90 %	82 %	↘	80 %	0.88	↘	0.82
Aca2009	Malt	stackeager	84 %	↗	86 %	81 %	↗	84 %	0.90	↗	0.95
		covnonproj	88 %	↗	90 %	85 %	↗	90 %	0.92	↗	1.00
		stacklazy	86 %	↗	90 %	83 %	↗	86 %	0.91	↗	0.95
	Mate	–	89 %	↗	92 %	85 %	↗	89 %	0.93	↗	0.95
	RBG	basic	83 %	→	83 %	79 %	↗	81 %	0.89	→	0.89
		standard	86 %	↘	79 %	83 %	↘	78 %	0.91	↘	0.89
		full	88 %	↘	86 %	84 %	→	84 %	0.91	↗	0.92
	Turbo	basic	80 %	↗	84 %	76 %	↗	84 %	0.86	↗	0.90
		standard	83 %	↗	87 %	79 %	↗	86 %	0.88	↗	0.92
		full	84 %	↗	89 %	80 %	↗	87 %	0.89	↗	0.94

Table 7: Comparison between overall attachment measures and attachment measures with only OOV tokens as basic population, for parser outputs based on gold-standard part-of-speech and morphological features as input. Arrows indicate the direction of change from overall to OOV.

Text	Parser	model type (resp. order)	UAS		UAS OOV	LAS		LAS OOV	deprel		deprel OOV
HDTtest	Malt	stackeager	84 %	↘	82 %	80 %	↘	78 %	0.87	↘	0.82
		covnonproj	88 %	↘	84 %	84 %	↘	80 %	0.89	↘	0.82
		stacklazy	86 %	↘	83 %	82 %	↘	78 %	0.88	↘	0.82
	Mate	–	88 %	↘	85 %	84 %	↘	79 %	0.89	↘	0.80
	RBG	basic	82 %	↘	80 %	78 %	↘	73 %	0.84	↘	0.75
		standard	86 %	↘	82 %	81 %	↘	74 %	0.86	↘	0.76
		full	87 %	↘	84 %	81 %	↘	76 %	0.87	↘	0.77
	Turbo	basic	81 %	↘	78 %	76 %	↘	69 %	0.82	↘	0.71
		standard	86 %	↘	83 %	80 %	↘	74 %	0.85	↘	0.75
		full	86 %	↘	83 %	80 %	↘	74 %	0.85	↘	0.75
Lit2009	Malt	stackeager	87 %	→	87 %	82 %	↘	76 %	0.89	↘	0.79
		covnonproj	90 %	↗	91 %	84 %	↘	83 %	0.88	↘	0.84
		stacklazy	88 %	↘	85 %	83 %	↘	75 %	0.88	↘	0.79
	Mate	–	89 %	↗	92 %	84 %	↘	81 %	0.89	↘	0.83
	RBG	basic	86 %	↘	81 %	80 %	↘	71 %	0.85	↘	0.73
		standard	89 %	↘	85 %	83 %	↘	73 %	0.87	↘	0.76
		full	89 %	↘	87 %	83 %	↘	75 %	0.87	↘	0.79
	Turbo	basic	85 %	↗	91 %	78 %	↘	71 %	0.83	↘	0.73
		standard	88 %	↗	89 %	80 %	↘	72 %	0.86	↘	0.75
		full	89 %	↗	91 %	81 %	↘	72 %	0.86	↘	0.75
Lit1850	Malt	stackeager	78 %	↘	77 %	73 %	↘	69 %	0.84	↘	0.76
		covnonproj	84 %	↘	79 %	79 %	↘	73 %	0.86	↘	0.76
		stacklazy	82 %	↘	77 %	76 %	↘	68 %	0.84	↘	0.70
	Mate	–	87 %	↘	83 %	82 %	↘	76 %	0.88	↘	0.77
	RBG	basic	82 %	↘	77 %	76 %	↘	67 %	0.84	↘	0.69
		standard	85 %	↘	82 %	79 %	↘	70 %	0.85	↘	0.73
		full	86 %	↘	82 %	80 %	↘	72 %	0.85	↘	0.74
	Turbo	basic	80 %	↘	76 %	72 %	↘	61 %	0.81	↘	0.63
		standard	84 %	↗	86 %	76 %	↘	71 %	0.83	↘	0.73
		full	84 %	↗	87 %	76 %	↘	71 %	0.83	↘	0.73
Aca2009	Malt	stackeager	82 %	↗	86 %	79 %	↗	83 %	0.89	↗	0.92
		covnonproj	85 %	↗	89 %	82 %	↗	87 %	0.90	↗	0.97
		stacklazy	85 %	↗	89 %	81 %	↗	84 %	0.90	↗	0.92
	Mate	–	85 %	↗	89 %	81 %	↗	84 %	0.89	↗	0.94
	RBG	basic	80 %	↘	75 %	77 %	↘	73 %	0.86	↘	0.83
		standard	84 %	↘	75 %	80 %	↘	73 %	0.88	↘	0.86
		full	84 %	↘	76 %	80 %	↘	75 %	0.88	↘	0.86
	Turbo	basic	77 %	↗	81 %	73 %	↗	79 %	0.84	↗	0.90
		standard	81 %	↗	87 %	76 %	↗	84 %	0.86	↗	0.92
		full	82 %	↗	87 %	77 %	↗	84 %	0.86	↗	0.94

Table 8: Comparison between overall attachment measures and attachment measures with only OOV tokens as basic population, for parser outputs based on computed part-of-speech and morphological features as input. Arrows indicate the direction of change from overall to OOV.