Document Parsing: Towards Realistic Syntactic Analysis

Rebecca Dridan♠, Stephan Oepen♠♡
♠ University of Oslo, Department of Informatics
♡ Potsdam University, Department of Linguistics
{rdridan|oe}@ifi.uio.no

Abstract

In this work we take a view of syntactic analysis as processing 'raw', running text instead of idealised, pre-segmented inputs—a task we dub *document parsing*. We observe the state of the art in sentence boundary detection and tokenisation, and their effects on syntactic parsing (for English), observing that common evaluation metrics are ill-suited for the comparison of an 'end-to-end' syntactic analysis pipeline. To provide a more informative assessment of performance levels and error propagation throughout the full pipeline, we propose a unified evaluation framework and gauge document parsing accuracies for common processors and data sets.

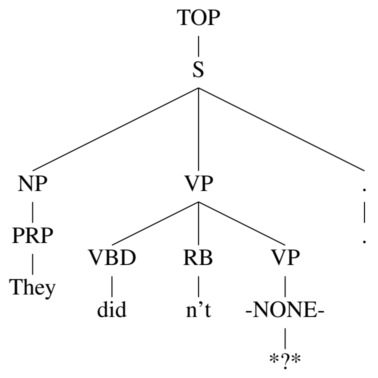
1 Introduction

The term *parsing* is used somewhat ambiguously in Natural Language Processing. In its 'pure', isolated interpretation, parsing maps from a sequence (or maybe a lattice) of token objects to syntactic analyses, say trees; from a more 'practical' point of view, however, parsing is also used to refer to a complete pipeline for syntactic analysis, starting with just running text (i.e. a string of characters) as its input. The two interpretations are of course closely related, but the underlying difference of perspective seems closely correlated with the distinction between parser *developers* vs. parser *users*. The parsing research literature and most contrastive benchmarking have their focus on parsing in isolation; at the same time, parsing software is most commonly applied to 'raw' string inputs. In this work, we take the practical, document parsing point of view and seek to inform parser users about configuration choices in a complete syntactic analysis pipeline, as well as about what they can expect in terms of end-to-end performance.

Parser evaluation, complementary to its utility in system development and scholarly comparison, has as one of its central purposes an aim of providing an indication of how a specific system or configuration might be expected to perform on unseen text. Genre and domain variation have been observed as important factors in predicting parser performance (Gildea, 2001; Zhang & Wang, 2009; Petrov & McDonald, 2012), but more fundamental issues related to pre-processing of parser inputs have largely been ignored.[1] That is, parser evaluation so far has been predominantly performed on perfectly segmented inputs. Indeed, the defacto standard for evaluating phrase structure parsers is the PARSEVAL metric (Black et al., 1991, as implemented in the `evalb` tool), which assumes that both the gold standard and the parser output use the same tokenisation—an unrealistic idealisation from the user point of view.

Recent research (Dridan & Oepen, 2012; Read et al., 2012) has shown that many standard NLP systems, and even tools specialised for sentence and token segmentation perform well below 100% on these fundamental tasks. However, the effects of errors so early in the pipeline on end-to-end parser accuracy are currently unexplored. The goal of this paper is to discover and document these effects, and also to promote a complementary tradition of parsing real documents, rather than the idealised, pre-segmented token sequences used in parser development. For these purposes, we propose a unified evaluation perspective applicable across all sub-tasks of syntactic analysis (§ 2) and contrast it with earlier work on parser evaluation under ambiguous tokenisation (§ 3). In § 4 and § 5 we apply this framework to a number of parsing pipelines built from state-of-the-art components and demonstrate that differences in pre-processing can yield large differences in parser performance, varying across domains and parsers—larger in fact than incre-

[1] Until recently, it was in fact difficult to obtain raw, unsegmented text for the most widely used English parser evaluation data, Wall Street Journal articles from 1989.



⟨0, 12, TOP⟩ ⟨0, 12, S⟩ ⟨0, 4, NP⟩
⟨5, 11, VP⟩ ⟨11, 11, VP⟩

Figure 1: Parse tree and span labels for *They didn't.*

mental PARSEVAL improvements frequently reported for 'pure' statistical parsing in the past decade.

2 Robust Evaluation of Syntactic Analysis

Parsing gold-standard token sequences into trees, common training and testing data, and a standard evaluation method have given the research community a well-defined and stable task over the past two decades that has enabled great advances in statistical parsing. In order to evaluate document parsing, however, we need to generalise the evaluation method somewhat.

PARSEVAL evaluates spans of text defined by token index. If the tokens are no longer considered as given, this definition is invalid. We take the obvious step and opt for character-based span indexing, with the slight twist of using inter-character positions from the raw text, rather than character counts. This allows greater flexibility and precision in describing spans. Figure 1 shows a parse tree and the phrase structure spans extracted from it, assuming this sentence started at character position 0 in the document.

While character-based token indexing is a simple concept, it is complicated by the fact that both treebanks and parsers 'normalise' raw text in a variety of ways, including Unicode/ASCII mappings, quote disambiguation and bracket escaping. In order to calculate the character spans for a token and be able to match to a system output that might use a different normalisation, we have implemented a tool that aligns between the tokens of an analysis and the raw input to the parsing process. Since we are not assuming gold sentence segmentation, this requires aligning full documents, making a simple implementation of string edit distance, for example, intractable. Instead, we have implemented the space-efficient longest common subsequence algorithm of Myers (1986), and calculate the shortest edit script with consideration of the most commonly accepted mappings. This enables an efficient alignment that is even robust to occasional missing analyses (due to parse failure).

Phrase structure is not the only annotation that can be represented as a labelled span. Many standard NLP tasks, including part-of-speech (PoS) tagging, tokenisation, and sentence segmentation can also be viewed this way. For PoS tagging, the parallel is obvious. Tokenisation and sentence segmentation have previously been evaluated in a variety of manners (Palmer & Hearst, 1997; Kiss & Strunk, 2006; Read et al., 2012), but are also amenable to being seen as span labelling, allowing us to use a consistent generalised evaluation framework. Thus, we represent each annotation as a triple consisting of span start and end positions, together with a label. This is particularly useful for our present purpose of evaluating the effect of these pre-processing tasks on phrase structure parsing. Figure 2 shows all such triples that can be extracted from our running example.[2]

3 Previous Work

A character-based version of `evalb`, dubbed SParseval, was earlier used for parser evaluation over the output from speech recognisers, which cannot be expected (or easily forced) to adhere to gold standard sentence and token segmentation (Roark et al., 2006). As word recognition errors can lead to very different parse yields, this approach required a separate, external word alignment step, far more complicated than our in-built alignment stage.

Parsing morphologically rich languages is another area where expecting gold standard parse yields is unrealistic. In this case, parse yields are often morpheme sequences, ambiguously derived from the input strings. Tsarfaty et al. (2012) propose an eval-

[2]Note that, at this point, we include several elements of the gold-standard annotation of Figure 1 that are commonly suppressed in parser evaluation, notably the root category TOP, the empty node at position $\langle 11, 11 \rangle$, and the final punctuation mark. In § 4 below, we return to some of these and reflect on the common practice of ignoring parts of the gold standard, against our goal of 'realistic' parser evaluation.

1:	⟨0, 4, POS:PRP⟩	2:	⟨0, 4, TOK⟩
3:	⟨5, 8, POS:VBD⟩	4:	⟨5, 8, TOK⟩
5:	⟨8, 11, POS:RB⟩	6:	⟨8, 11, TOK⟩
7:	⟨11, 11, POS:-NONE-⟩	8:	⟨11, 11, TOK⟩
9:	⟨11, 12, POS:.⟩	10:	⟨11, 12, TOK⟩
11:	⟨0, 12, TOP⟩	12:	⟨0, 12, S⟩
13:	⟨0, 4, NP-SBJ⟩	14:	⟨5, 11, VP⟩
15:	⟨11, 11, VP⟩	16:	⟨0, 12, SENT⟩

Figure 2: All triples from the tree in Figure 1.

uation metric (TEDEVAL) based on tree edit distance for Hebrew, rejecting character-based evalb as insufficient because it cannot (a) evaluate morphemes unrealised in the surface form; or (b) enforce only yields that break at whitespace and correspond to a path through the lattice of the morphological analyser. For languages like English, where the raw string is conventionally a concatenation of the parse yield, observation (a) is not an issue. However, we note that zero-length spans, made possible by our decision to record spans in terms of inter-character positions, can preserve linear precedence without requiring explicit realisation in the raw text. As for the second objection of Tsarfaty et al. (2012), that is specific to their situation of allowing ambiguous morphological segmentation only within pre-segmented tokens. In our more general case, we consider enumerating all possible tokenisations of a sentence (or document) neither practical or desirable, nor do we wish to enforce token breaks at whitespace, since different tokenisation conventions such as those described by Fares et al. (2013) allow mid-token spaces.

4 Experimental Setup

To evaluate end-to-end parsing, we selected three commonly used phrase structure parsers: the Charniak and Johnson reranking parser (C&J; Charniak & Johnson, 2005), the Berkeley parser (Petrov, Barrett, Thibaux, & Klein, 2006), and the Stanford parser (Klein & Manning, 2003). For each parser, we used the recommended pre-trained English model, and mostly default settings.[3] All parsers tokenise their input by default and the Stanford parser also includes a sentence splitting component. Since both the C&J and Berkeley parsers require sentence-segmented text, we pre-segmented for those parsers using `tokenizer`, the top-performing, lightweight, rule-based segmenter in the recent survey of Read et al. (2012).

For parser evaluation, the standard test data is Section 23 of the venerable WSJ portion of the Penn Treebank (PTB, Marcus et al., 1993). In addition to this data, we also use the full Brown portion of the PTB (often used for experiments in domain variation) and the relatively new English Web Treebank (EWTB, LDC #2012T13), which comprises several web genres. To evaluate *document parsing*, we start from the raw, running text. In the case of the EWTB this was provided with the treebank; for the other corpora, we use the raw texts distributed by Read et al. (2012). Corpora sizes are included in Table 1.

Our reference implementation for evaluation instantiates the triple-based framework outlined in § 2 above. For compatibility with much previous work, our tool supports the same configuration options as evalb, to discard tree nodes or consider different labels as equivalent. Somewhat ambivalently (since we experience these 'tweaks' as obstacles to transparency and replicability), we mimic the standard practice of discarding triples labelled TOP (the root category) and -NONE- (empty elements, absent in most parser outputs), as well as truncating labels at the first hyphen (i.e. ignoring function tags); furthermore, we allow the commonly accepted equivalence of ADVP and PRT, even though we have been unable to trace the origins of this convention. However, we do not follow the convention of ignoring tokens tagged as punctuation, for two reasons. First, Black et al. (1991) suggested ignoring punctuation for independently developed, hand-built parsers, to reduce variation across linguistic theories; in evaluating statistical parsers trained directly on PTB structures, however, it is natural to include syntactic information related to punctuation in the evaluation—as is common practice in data-driven dependency parsing since the 2007 CoNLL Shared Task. Second, the evalb technique of identifying punctuation nodes merely by PoS tags is problematic in the face of tagging errors; and even if one were to 'project' gold-standard PoS tags onto parser outputs, there would be ambiguity about how to count erroneous labels involving punctuation spans.

[3]We used the `-accurate` setting for the Berkeley parser, to improve coverage on our out-of-domain data. Full configuration details and other background, including an open-source implementation of our evaluation framework will be distributed through a companion website.

Corpus	# Gold Sentences	# Gold Tokens	C&J		Berkeley		Stanford	
			Gold	Auto	Gold	Auto	Gold	Auto
Brown	13919	234375	82.7	81.2	82.2	82.3	77.1	77.7
EWTB	16622	254830	78.3	74.8	76.2	76.5	73.6	71.6
WSJ_{23}	2416	56684	90.9	86.4	89.8	88.4	85.4	83.8

Table 1: Phrase structure accuracy (labelled F_1) for three parsers (Charniak and Johnson, Berkeley and Stanford) over three different data sets, first using gold sentence segmentation and tokenisation, and then in the default string-input mode for each parser. Since only the Stanford parser performs sentence segmentation, the `tokenizer` sentence splitter was used to pre-process running text for the other two parsers.

	Sentences		Tokens							
Corpus	Tokenizer	Stanford	C&J		Berkeley		Stanford		REPP	
	F_1	F_1	F_1	SA	F_1	SA	F_1	SA	F_1	SA
Brown	96.0	95.7	98.3	77.0	99.5	85.8	99.5	86.1	99.6	86.5
EWTB	85.2	70.3	96.9	77.8	97.3	83.1	97.8	85.9	97.3	81.4
WSJ_{23}	94.4	93.5	99.1	86.8	99.6	94.8	99.6	94.9	99.9	98.0

Table 2: Preprocessing accuracy: Sentence segmentation is evaluated as the F_1 over sentence spans for `tokenizer` and the Stanford parser. Token accuracy is reported both as F_1 and sentence accuracy (SA), the percentage of sentences with completely correct tokenisation, for each of the parser-internal tokenisers, plus the stand-alone tokeniser REPP.

5 Experimental Results

End-to-end results of our first attempt at document parsing for each parser are shown in Table 1 (**Auto** columns), alongside the results of parsing idealised inputs with gold sentence and token segmentation (**Gold**). Accuracy is in terms of F_1 over phrase structure triples. The drop in F_1 from earlier published results on WSJ_{23} with gold tokens is due to our inclusion of punctuation and evaluation of all sentences, regardless of length.

We see a wide variation in how automatic preprocessing affects parser accuracy for the different parsers and domains. Focussing on WSJ_{23}, the in-domain set for all parser models, we see a drop in parser accuracy for all parsers when moving away from gold tokenisation. However, the size of the drop varies across parsers, with the C&J parser particularly affected. Indeed, with automatic pre-processing the Berkeley parser is now the more accurate for this data set. Over the out-of-domain sets, the C&J parser again loses accuracy switching from gold tokens, but the drop is not so severe, even though we would expect more pre-processing errors in, for example, the web text of EWTB.

For the Brown data set, the numbers show a slight *increase* in accuracy for both Berkeley and Stanford parsers when using automatically processed input, an unexpected effect. Berkeley again out-performs the C&J parser when not using gold tokens.

The low scores on EWTB are not surprising, since none of the parsers were tuned to this quite different genre, but the relative difference between idealised and document parsing is unexpected, giving how difficult one might expect this text to be for sentence segmentation and tokenisation. Again, we see that the Berkeley parser has actually improved in accuracy with automatic pre-processing, but the Stanford parser shows a similar drop to that it produced over WSJ_{23}.

In order to drill down to the reasons behind some of these variations, Table 2 shows the accuracy of the sentence segmentation and tokenisation used in the automatic configurations above.

Looking at sentence segmentation in Table 2, we see that accuracy is much lower over EWTB, reflecting the more informal nature of these texts, particularly with regards to capitalisation and punctuation. Our numbers are lower than in Read et al. (2012), due to differences in the evaluation method,[4] but we also see that `tokenizer` is the more accurate sentence splitter. The Stanford sentence segmentation is mod-

[4]While they evaluated only boundary points (i.e. sentence end points), we are interested in the effect of incorrect sentence segmentation on parsing and so evaluate the full sentence span (i.e. start and end points jointly). Thus, one incorrect sentence boundary point can lead to two incorrect sentence spans.

erately less accurate for the edited text, but radically under-segments the messier web text, returning 12914 sentences, compared to 14954 from `tokenizer`.

For tokenisation accuracy, there's very little difference between Berkeley and Stanford, which is not surprising, since the Berkeley parser tokeniser was built using the Stanford code. The in-built tokeniser in the C&J parser however is significantly less accurate. While the drop in F_1 doesn't seem too large, examining the sentence accuracy shows a much bigger effect. This echoes the findings of Dridan and Oepen (2012) which demonstrate that the built-in tokeniser of the C&J parser is well below state-of-the-art. For completeness, we also looked at their best performing tokeniser (REPP) on `tokenizer`-segmented sentences, adding those token accuracy numbers to Table 2. Here we see that relative performance of the better tokenisers varies with domain. As Dridan and Oepen (2012) found, REPP outperforms Stanford over Wall Street Journal text, but the performance difference over Brown is slight, and on EWTB, the advantage is to the Stanford tokeniser.

We re-parsed using the best tokeniser and sentence segmenter for each domain, with the results shown in Table 3. By comparing back to Table 1, we can see that there are a few times that the best pre-processing accuracy doesn't lead to the best parsing results, but the differences are slight. While the `tokenizer`+REPP combination gave the best performance for all parsers on WSJ_{23}, the results over Brown are surprising. The best input for each parser was the default, except for the C&J parser, where `tokenizer`+REPP led to the best results. The automatic tokenisers led to equal *or better* parse accuracy than using gold tokens. Our hypothesis to explain this phenomenon is that subtle differences in the tokenisation of the Brown corpus and the EWTB, compared to the WSJ portion of the PTB, confuse the parsers, which have been tuned not just to the WSJ text type, but also to the idiosyncrasies of its annotation. Giving the parsers 'more familiar' inputs can thus lead to more accurately bracketed analyses.

Sentence segmentation is still a major impact on the comparatively low parse accuracies on EWTB. Switching to `tokenizer` improved Stanford parser accuracy by almost 1 point, but further experiments using gold sentence boundaries with automatic tokenisation showed a boost of 2–4 points F_1, leaving further room for improvement there.

Corpus	**C&J**		**Berkeley**		**Stanford**	
	Gold	**Auto**	**Gold**	**Auto**	**Gold**	**Auto**
Brown	82.7	82.7	82.2	82.2	77.1	77.6
EWTB	78.3	77.7	76.2	76.3	73.6	72.4
WSJ_{23}	90.9	89.9	89.8	88.8	85.4	84.5

Table 3: Final document parsing phrase structure accuracies, showing the results of using the best combination of sentence segmentation and tokenisation for each domain. Gold tokenisation F_1numbers are repeated for easy comparison.

Overall, our results show that document parsing is a viable task, leading to parser accuracies only slightly below the state of the art over gold tokens for the (potentially overfitted) WSJ_{23}, given the right preprocessors. Furthermore, by actually evaluating the end-to-end pipeline, as well as the performance, in-situ, of the various preprocessors we have discovered that parser accuracy can actually be improved by matching the tokenisation to the expectations of the parser, rather than merely using the tokenisation of the treebank. This subtle tuning of annotation style, rather than domain could even help explain previous cross-domain results (McClosky, Charniak, & Johnson, 2006) that showed better accuracy from *not* using labelled Brown data for parser training.

6 Conclusions and Outlook

Through this work, we hope to call the syntactic analysis research community to arms and initiate a shift of emphasis towards practical use cases and *document parsing*—with the aim of mitigating the risks of overestimating parser performance and over-tuning to individual treebank idiosyncrasies. Our triple-based proposal for Robust Evaluation of Syntactic Analysis synthesises earlier work and existing metrics into a uniform framework that (a) encompasses the complete analysis pipeline; (b) enables comparison between analyses that use different normalisation conventions of the same text; and (c) allows one to quantify performance levels of individual sub-tasks as well as degrees of error propagation. Furthermore, we demonstrate how this setup supports 'mixing and matching' of gold-standard and system outputs, to isolate the effects of individual sub-tasks on end-to-end performance, which can bring to light unexpected effects.

Our evaluation framework is supported by an open-source reference implementation, with configuration options very similar to `evalb`, that can be used as a plug-in replacement too for evaluation of the complete syntactic analysis pipeline.[5] Immediate future work on this tool is planned to also allow evaluation of dependency analysis under ambiguous tokenisation by also evaluating tuples consisting of two character-defined sub-spans, plus a dependency label.

Acknowledgments

We are grateful to our colleagues at the Oslo Language Technology Group for many helpful discussions and suggestions, as well as to our three anonymous reviewers for insightful comments. This work is in part funded by the Norwegian Research Council through its WeSearch project. Large-scale experimentation is made possible through access to the ABEL high-performance computing facilities at the University of Oslo, and we are grateful to the Scientific Computing staff at UiO, as well as to the Norwegian Metacenter for Computational Science, and the Norwegian tax payer.

[5]Please see `http://www.delph-in.net/resa` for further information on how to obtain and run our tool.

References

Black, E., Abney, S., Flickinger, D., Gdaniec, C., Grishman, R., Harrison, P., … Strzalkowski, T. (1991). A procedure for quantitatively comparing the syntactic coverage of English grammars. In *Proceedings of the workshop on speech and natural language* (p. 306 – 311). Pacific Grove, USA.

Charniak, E., & Johnson, M. (2005). Coarse-to-fine n-best parsing and MaxEnt discriminative reranking. In *Proceedings of the 43rd Meeting of the Association for Computational Linguistics* (p. 173 – 180). Ann Arbor, MI, USA.

Dridan, R., & Oepen, S. (2012). Tokenization. Returning to a long solved problem. A survey, contrastive experiment, recommendations, and toolkit. In *Proceedings of the 50th Meeting of the Association for Computational Linguistics* (p. 378 – 382). Jeju, Republic of Korea.

Fares, M., Oepen, S., & Zhang, Y. (2013). Machine learning for high-quality tokenization. Replicating variable tokenization schemes. In *Computational linguistics and intelligent text processing* (p. 231 – 244). Springer.

Gildea, D. (2001). Corpus variation and parser performance. In *Proceedings of the 2001 Conference on Empirical Methods in Natural Language Processing* (p. 167 – 202). Pittsburgh, USA.

Kiss, T., & Strunk, J. (2006). Unsupervised multilingual sentence boundary detection. *Computational Linguistics*, *32*(4), 485 – 525.

Klein, D., & Manning, C. D. (2003). Accurate unlexicalized parsing. In *Proceedings of the 41st Meeting of the Association for Computational Linguistics* (p. 423 – 430). Sapporo, Japan.

Marcus, M., Santorini, B., & Marcinkiewicz, M. A. (1993). Building a large annotated corpora of English: The Penn Treebank. *Computational Linguistics*, *19*, 313 – 330.

McClosky, D., Charniak, E., & Johnson, M. (2006). Reranking and self-training for parser adaptation. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th Meeting of the Association for Computational Linguistics* (p. 337 – 344). Sydney, Australia.

Myers, E. W. (1986). An O(ND) difference algorithm and its variations. *Algorithmica*, *1*(1-4), 251 – 266.

Palmer, D. D., & Hearst, M. A. (1997). Adaptive multilingual sentence boundary disambiguation. *Computational Linguistics*, *23*(2), 242 – 267.

Petrov, S., Barrett, L., Thibaux, R., & Klein, D. (2006). Learning accurate, compact, and interpretable tree annotation. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th Meeting of the Association for Computational Linguistics* (p. 433 – 440). Sydney, Australia.

Petrov, S., & McDonald, R. (2012). *Overview of the 2012 Shared Task on Parsing the Web.* Notes of the First Workshop on Syntactic Analysis of Non-Canonical Language. Montreal, Canada.

Read, J., Dridan, R., Oepen, S., & Solberg, L. J. (2012). Sentence boundary detection: A long solved problem? In *Proceedings of the 24th International Conference on Computational Linguistics.* Mumbai, India.

Roark, B., Harper, M., Charniak, E., Dorr, B., Johnson, M., Kahn, J. G., … Yung, L. (2006). SParseval: Evaluation metrics for parsing speech. In *Proceedings of the 5th International Conference on Language Resources and Evaluation.* Genoa, Italy.

Tsarfaty, R., Nivre, J., & Andersson, E. (2012). Joint evaluation of morphological segmentation and syntactic parsing. In *Proceedings of the 50th Meeting of the Association for Computational Linguistics* (p. 6 – 10). Jeju, Republic of Korea.

Zhang, Y., & Wang, R. (2009). Cross-domain dependency parsing using a deep linguistic grammar. In *Proceedings of the 47th Meeting of the Association for Computational Linguistics* (p. 378 – 386). Suntec, Singapore.