BERT-Proof Syntactic Structures: Investigating Errors in Discontinuous Constituency Parsing

Maximin Coavoux
Univ. Grenoble Alpes, CNRS, Grenoble INP, LIG
maximin.coavoux@univ-grenoble-alpes.fr

Abstract

The combined use of neural scoring systems and BERT fine-tuning has led to very high results in many natural language processing (NLP) tasks. These high results raise two important questions about the contribution and the limitations of pretrained-language models: (i) what are the remaining errors in the best-performing systems? (ii) what are the types of test examples where pretrained language models help the most? In this paper, we investigate both questions for the task of English discontinuous constituency parsing on the Penn Treebank, for which recent models obtain close to 95 F_1 score. To do so, we propose two methods for automatically analysing the errors of discontinuous parser. First, we annotate and release a test-suite focused on the syntactic phenomena responsible for discontinuities in the Penn Treebank, enabling us to obtain a per-phenomenon evaluation of a parser's output. Second, we extend the Berkeley Parser Analyser — a tool that classifies parsing errors according to predefined structural patterns —, to discontinuous trees. We apply both methods to characterize errors of a state-of-the-art transition-based discontinuous parser, and to provide an overview of the contribution of BERT to this task.

1 Introduction

Discontinous constituency trees are phrase-based syntactic representations where the constraint stating that a single phrase must yield a continuous sequence of tokens is lifted. Such representations are well-suited for modelling long-range dependencies, that typically arise for some syntactic phenomena, such as extractions or scrambling. For example, Figure 1 presents a discontinous VP modelling the relationship between the verb *want* and its extracted complement *How many*.

In constituency treebanks, these long range dependencies are sometimes represented with typed

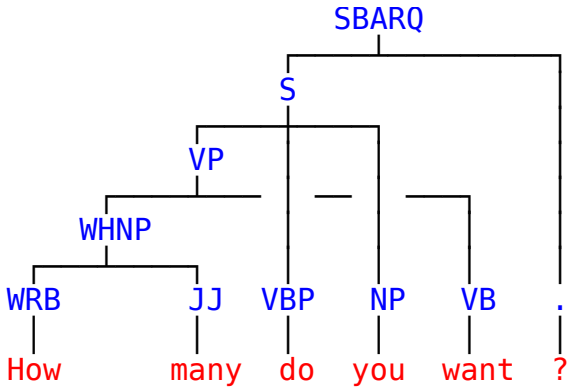


Figure 1: Discontinuous constituency tree.

empty categories (traces), coindexed with a displaced phrase (Marcus et al., 1993). However, projective parsers usually ignore them.[1] Indeed, the norm in Penn Treebank constituency parsing is to preprocess empty categories out of the corpus, leaving out important linguistic information.

Formally, discontinuous constituency trees interpret as derivations from mildly context-sensitive grammar formalisms, such as linear context-free rewriting systems (Vijay-Shanker et al., 1987, LCFRS) or multiple context-free grammars (Seki et al., 1991, MCFG). As a result, exact parsing of discontinuous structures has high computational complexity. For example, CKY-style parsing of an LCFRS is $\mathcal{O}(n^{3f})$ in time (Kallmeyer, 2010), where f is the fan-out of the grammar: the maximum number of spans in a grammar rule.[2]

The current state of the art for English discontinuous constituency parsing on the Discontinuous Penn Treebank has reached 94.8% F_1 score (Corro, 2020) obtained by a span-based chart parser that combines a neural scoring system and pretrained contextualized embeddings (Devlin et al., 2019,

[1] Except for some work on parsing traces, e.g. Gabbard et al. (2006); Kummerfeld and Klein (2017).

[2] If $f = 1$, the grammar is equivalent to a CFG.

Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 3259–3272

BERT). However, such a high score can be misleading. Despite ensuring comparability across different parsers, the exclusive use of classical evaluation metrics (F-score, precision, recall, as is standard) is hard to interpret and does not disclose information about the syntactic capabilities of a parser. In discontinuous parsing, the standard evaluator `discodop`[3] (van Cranenburgh et al., 2016) provides metrics that only focus on discontinuous constituents (discontinuous F-score, discontinuous precision, discontinuous recall). However, these scores aggregate information across many distinct syntactic phenomena.

In this paper, we propose to automatically analyse the errors of discontinuous English parsers in order to provide a fine-grained overview of their current limitations. To do so, we pursue two complementary approaches. First, we construct a test suite focused on 6 syntactic phenomena responsible for the discontinuities in the Discontinuous Penn Treebank (Evang, 2011). Second, we adopt an error-correction based approach: we search for a sequence of error-correcting tree modifications that lead from the predicted tree to the gold tree, and classify the sequence of tree modifications based on structural patterns. This is a direct extension of Berkeley Parser Analyser (Kummerfeld et al., 2012) to English discontinuous parsing.

A secondary motivation for this work is to characterize the contribution of BERT to discontinuous constituency parsing. An active current line of research consists in assessing the syntactic knowledge learned by language models (Linzen et al., 2016; Marvin and Linzen, 2018; Gulordava et al., 2018), including those with structural supervision (Kuncoro et al., 2018; Wilcox et al., 2019; Hu et al., 2020). They usually do so by constructing test items: minimal pairs of sentences, such that one is grammatical and the other is not (thus isolating a single grammatical constraint). Then, they observe whether the language model assigns higher probability to the grammatical alternative. In these papers, the observation of the syntactic ability of the models is indirect. We argue that fine-grained evaluation methods will help comparing the syntactic capabilities of parsers when they have access to BERT or not, which will provide a complementary view to this line of research. Therefore, we apply both proposed error analyses methods to a state-of-the-art transition-based discontinuous parser in several settings: without pretraining, with fast-text embeddings (Mikolov et al., 2018a; Grave et al., 2018), with BERT pretraining.

In summary, we make the following contributions:

- We construct a test-suite for automating a fine grained evaluation of English discontinuous parsers on target phenomena.
- We extend the Berkeley parser analyser to deal with English discontinuous constituency trees.
- We use these two evaluation methods to characterize the errors of a neural discontinuous parser, trained in several pretraining settings.

We provide the test suite and the error analyser as supplementary material.

2 Related Work

To address the limitations of using exclusively an F-score to evaluate constituency parsers, prior work focused on alternative finer-grained evaluation methods. We review some of them, both from the projective and discontinuous constituency parsing litterature.

Manual error analysis For discontinuous constituency parsing, Evang (2011) performed manual error analysis by extracting discontinuous trees from the evaluation corpus, classifying them according to the phenomenon at the origin of the discontinuities, and manually checking if a PLCFRS chart-parser recognized them. Coavoux et al. (2019) used the same strategy to evaluate a neural transition-based discontinuous parser. However, manual error analysis is quite time-consuming and needs to be performed again for evaluating each new parser output. Thus, it is difficult to integrate it in an evaluation pipeline or to deploy it for many parsers.

Automatic error analysis Kummerfeld et al. (2012) introduced a method that consists in searching for a sequence of atomic tree-modifications (such as: inserting a node, removing a node, moving a node) that leads from a predicted constituency tree to the gold tree. Then, they classify the tree modifications according to predefined structural patterns, e.g., 'PP-attachment', 'NP-attachment', 'labelling error'. Their method led to identify most frequent patterns of error, and characterize the improvement obtained with techniques such as reranking. However, the structural patterns used to clas-

[3] https://github.com/andreasvc/disco-dop/

sify mistakes depend both on the language of the treebank and on its annotation strategies. Therefore, error patterns need to be designed when adapting the error analyser to another treebank (Kummerfeld et al., 2013). Moreover, their method and software do not handle discontinuous constituents, hence our proposal.

Targeted evaluation Another line of work on fine-grained parser evaluation focused on specific structures or phenomena. Ratnaparkhi et al. (1994) introduced a collection of English sentences with PP-attachment ambiguities, in order both to improve evaluation on this type of structure, and foster research on improving their resolution. Kübler et al. (2009) introduced a test suite for German that encompasses a wider range of syntactic structures (such as coordination of unlike constituents or extraposed relative clauses). However, they focus on projective constituency representations.

For discontinuous structures, Maier et al. (2014) released `discosuite`, a testsuite for German. They annotated a set of sentences from the Tiger corpus (Brants et al., 2004), with the syntactic phenomena responsible for the tree discontinuities. They released their annotations, such that researchers can run their parsers on the sentences and compute a per-phenomenon evaluation of the parser. To the best of our knowledge, such a test suite only exists for German. In this article, we introduce one for English, along with an evaluation script that provides per-phenomenon statistics.

We focus our analysis on English, since the resources we introduce are for this language. However, we also provide results on German using `discosuite`.

3 Test Suite Annotation

This section describes our methodology to annotate a set of discontinuous constituency trees with the syntactic phenomena responsible for the discontinuities. We first extract all discontinuous trees from the validation section of the discontinuous version of the Penn Treebank (Evang, 2011; Evang and Kallmeyer, 2011), except those for which the discontinuities are only due to punctuation attachment. We obtain 266 trees, which corresponds to 16% of the corpus. Then, we manually assign one or several categories from the following set, previously proposed for manual error analysis by Evang (2011) and reused by Coavoux et al. (2019). We provide an example sentence from the corpus for each category, with the main discontinuous constituent highlighted in bold:

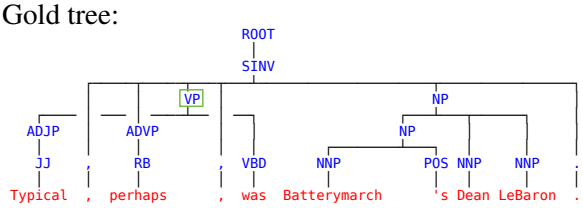
1. *wh*-extraction: *[. . .] the most recent period* ***for which*** *results were* ***broken out*** *[. . .]*
2. circumpositioned quotation: ***While Mayor Norman found the market's performance Monday reassuring,*** *he says,* ***he remains uneasy.***
3. fronted quotations: ***The proposed changes "all make a lot of sense to me,"*** *he* ***added****.*
4. *it*-extraposition: *While* ***it*** *is possible* ***that the Big Green initiative will be ruled unconstitutional*** *[. . .]*
5. discontinuous dependencies: *[. . .] provided little help for copper as* ***word*** *spread* ***that a three-month strike at the Highland Valley mine in British Columbia was about over*** *[. . .]*
6. subject-verb inversion: ***Added*** *another executive at a big bank:* ***"We were all a little goosey over the weekend trying to forecast what would happen Monday, but it's been very quiet.***

There are several subtypes of *wh*-extractions in the data: relative clauses, verbal adjunct clauses, complement clauses, indirect and direct questions. They all include a *wh* word among *how, when, which, that, where, what, why, whenever*. Circumpositioned and fronted phrases only include quotations, and systematically feature a speech verb, usually *says* or *said*. *It*-extrapositions feature an expletive *it* in the interpretation location of an extraposed clausal argument. The category of discontinuous dependencies contains other cases where a constituent is split by an intervening phrase. It mostly includes extraposed modifiers, such as the extraposed clause in example 5 above,

Not all occurrences of these phenomena result in a discontinuous tree (Evang, 2011). For example, a sentence containing both a fronted quotation and a subject-verb inversion will not result in a discontinuity. In some trees, there are also several occurrences of phenomena producing discontinuities. We release these annotations as a csv file, provided as supplementary material.

Per-phenomenon evaluation method In order to obtain a per-phenomenon evaluation of the predictions of discontinuous parsers, we first extract individual evaluations for each discontinuous tree, as provided by the standard evaluator for discontinuous parsing (van Cranenburgh et al., 2016, dis-

Gold tree:



Predicted tree:

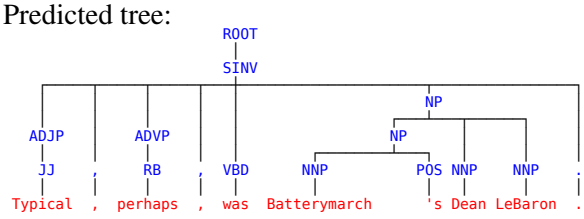


Figure 2: Error correction: add a missing node.

codop). These include the number of gold, correct, and incorrect discontinuous constituents, both in the labelled and unlabelled case.[4]

We consider that the annotated target phenomenon on the sentence is perfectly predicted if the sentence discontinuous F-score is 100, and partially predicted if it is > 0. As such, this evaluation is recall-oriented: we focus on how well the gold phenomena are predicted, but we do not take into account false positives (which would require us to assign a phenomenon to predicted trees with incorrect discontinuous constituents).

4 Error-Correction-Driven Analyser

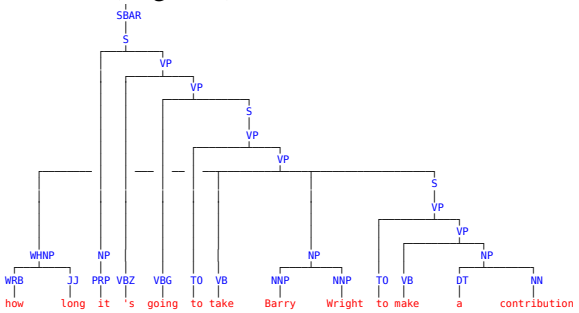
We now focus on automatically classifying errors according to structural patterns. To do so, we build on Kummerfeld et al. (2012) and proceed in two steps: (i) finding a sequence of atomic tree modifications that transforms a predicted tree to the corresponding gold tree; (ii) classifying steps in the transformation sequence according to predefined structural patterns. For step (i), we use a greedy search algorithm, that first corrects errors on discontinuous nodes, and then backoffs to Kummerfeld et al. (2012)'s method for projective error correction. Thus, we focus in this section on *discontinuous* error corrections.

We use 4 atomic tree modifications:

i change label of discontinuous node;
ii create a discontinuous node;
iii delete a discontinuous node;
iv move a node, resulting in a discontinuity.

[4]In the unlabelled case, we remove duplicate constituents (that correspond to unary rewrites) before evaluation as they are not interpretable. Therefore, it might happen that the labelled result is higher than the corresponding unlabelled one.

Gold tree (fragment):



Predicted tree (fragment):

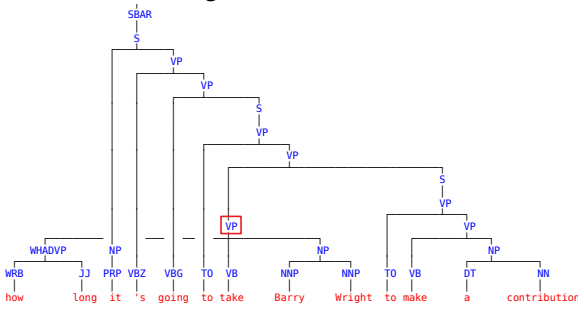


Figure 3: Error correction: remove an extra node.

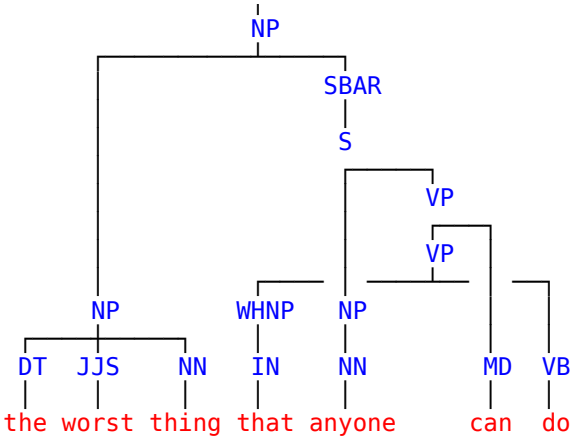
The creation of a discontinuous node (ii) consists in gathering several nodes with the same parent, attaching them as the children of a new node, which is attached in turn to their original parents. For example, in Figure 2, the parser missed a VP node. The correction consists in creating a discontinuous VP node with two children (ADJP and VBD nodes) and attaching it to the SINV node.

To delete a discontinuous node (iii), we simply attach its children to their grandparent node. For example, in Figure 3, the children of the highlighted VP to be deleted (lower part) will be attached to the higher VP. For both node creation and node deletion, the corrected tree has only a single different node with the original predicted tree.

Finally, the moving of a node involves reattaching a node to a different parent. For example, in Figure 4, the correction of the predicted tree will consist in attaching the WHNP to the lowest VP, resulting in two missing discontinuous constituents (both VPs, see gold tree). A side effect is that the moving will also result in a unary S constituent that should be deleted in another correction step.

In order to find a sequence of error-correcting modifications, we perform a greedy search. While there is a false positive or a false negative discontinuous constituent in the current tree, we try to apply actions (i-iv) in this order of priority. Modifications (i-iii) cannot introduce new errors, whereas

Gold tree (fragment):



Predicted tree (fragment):

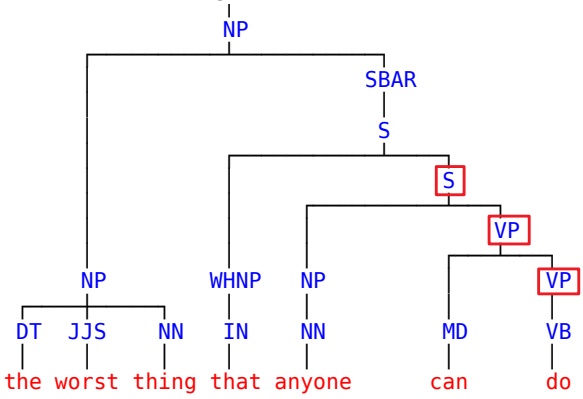


Figure 4: Error correction: move a node.

moving a node (iv) may do so in some cases.[5] We ensure that moving a node is only performed if the correction does not increase the gross number of errors by more than 1.

Once we have found the correction sequence, we classify errors according to patterns defined by Kummerfeld et al. (2012):

- PP attachment;
- NP internal structure;
- Modifier Attachment;
- Unary constituent;
- Different label;
- Clause attachement;
- Coordination;
- NP attachment;
- VP attachment.

Once all discontinuous errors are corrected, we run the original code from Kummerfeld et al. (2012) to compute statistics about projective constituent mistakes.

[5]The effect of creation/deletion of a single node is purely local, whereas the moving of the node may impact many other nodes. For example in Figure 4, the moving changes the yield of 3 nodes.

5 Parser

We use a Python reimplementation of the parser described by Coavoux et al. (2019), augmented with a mechanism to integrate and fine-tune BERT (Devlin et al., 2019). We release our code with pretrained models for replication purposes.[6]

The parser is based on a simple transition system (ML-GAP) that features the GAP action (Coavoux and Crabbé, 2017) to construct discontinuous constituents, and separates structural and labelling actions (Cross and Huang, 2016). The scoring system has two submodules:

- A sentence encoder that constructs contextualized embeddings for each token and is run before parsing;
- A feed-forward network that predicts the next action from the contextualized embeddings of tokens extracted from specific positions in the parsing configuration.

In the remainder of the paper, we call ML-GAP the baseline parser that has only access to the training corpus and has no pretrained parameters, ML-GAP+FT, when it has access to fasttext pretrained embeddings (Mikolov et al., 2018b), and ML-GAP+BERT, the parser that uses the `bert-base-cased` pretrained language model to compute token representations and finetunes it.

Token and sentence encoder The parsers differ in the way they represent the tokens $(w_1, w_2, \dots w_n)$ in a sentence. The ML-GAP parser computes character-based word embeddings with a character bi-LSTM: $(\mathbf{c}_1, \dots \mathbf{c}_n)$, where $\mathbf{c}_i = \text{bi-LSTM}(w_i)$, and concatenates them to word embeddings: $([\mathbf{c}_1, \mathbf{w}_1], \dots [\mathbf{c}_n, \mathbf{w}_n])$. The ML-GAP+FT parser replaces learned word embeddings by (frozen) fast-text embeddings. The ML-GAP+BERT parser also uses a character bi-LSTM, but its output is concatenated with the contextualized embeddings from BERT: $([\mathbf{c}_1, \mathbf{b}_1], \dots [\mathbf{c}_n, \mathbf{b}_n])$, where $(\mathbf{b}_1, \mathbf{b}_2, \dots, \mathbf{b}_n)$ is the output of the last layer BERT for the corresponding tokens. When BERT segments a token into several subtokens, we use the vector corresponding to the first subtoken. Alternative methods are available (using the last subtoken or an aggregation of the subtoken vectors) but they do not seem to have an effect on parsing (Kitaev et al., 2019).

[6]https://gitlab.com/mcoavoux/mtgpy-release-findings-2021, also archived at http://doi.org/10.5281/zenodo.4775955

Then, the token embeddings are fed to a bi-LSTM sentence encoder, as usually done in parsing (Stanojević and Alhama, 2017; Coavoux and Cohen, 2019; Corro, 2020; Stanojević and Steedman, 2020). In preliminary experiments, we alternatively used a self-attentive encoder (Vaswani et al., 2017), as done successfully in recent work in projective constituency parsing (Kitaev and Klein, 2018; Kitaev et al., 2019). However, it proved hard to optimize (high variance across experiments) and did not obtain better results than a bi-LSTM.

System	Dev		Test	
	F	DF	F	DF
Fully supervised				
$^{\alpha}$Evang and Kallmeyer (2011) < 25, gold POS			79	
$^{\alpha}$van Cranenburgh and Bod (2013) ≤ 40	85.2		85.6	
$^{\alpha}$van Cranenburgh et al. (2016) ≤ 40	86.9		87	
$^{\delta}$Corro et al. (2017)			89.2	
$^{\beta}$Coavoux and Cohen (2019)	91.4	70.9	90.9	67.3
$^{\beta}$Coavoux et al. (2019)	91.2	72.0	91.0	71.3
$^{\gamma}$Corro (2020)			**92.7**	64.2
$^{\gamma}$Stanojević and Steedman (2020)			90.5	67.1
$^{\alpha}$Ruprecht and Mörbitz (2021)			90.1	72.9
$^{\beta}$This work: ML-GAP	**92.0**	**75.9**	91.4	**74.4**
Semi-supervised (Pretrained embeddings)				
$^{\gamma}$Corro (2020)			**92.9**	64.9
$^{\alpha}$Ruprecht and Mörbitz (2021)			91.8	76.1
$^{\beta}$This work: ML-GAP+FT	92.7	78.1	92.3	**76.5**
Semi-supervised (Bert-base)				
$^{\gamma}$Corro (2020)			**94.8**	68.9
$^{\delta}$Vilares and Gómez-Rodríguez (2020)			91.7	49.1
$^{\alpha}$Ruprecht and Mörbitz (2021)			93.3	80.5
$^{\beta}$This work: ML-GAP+BERT	**95.0**	**85.8**	**95.0**	**82.5**

Table 1: Results on the Discontinuous Penn Treebank. DF: discontinuous F-score. $^{\alpha}$Grammar-based, $^{\beta}$transition-based, $^{\gamma}$chart-based, $^{\delta}$other neural systems.

Action scorer and features We use two distinct feed-forward networks to score respectively structural actions (SHIFT, MERGE, GAP) and labelling actions (NO-LABEL, {LABEL-X | X is a non-terminal}). They both have an identical architecture and only differ in the number of units in the output layer. We use a single hidden layer with a tanh activation. We apply dropout to its input, and layer normalization (Ba et al., 2016) to the hidden layer. We use a softmax normalization to compute scores for possible output labels.

The choice of the structural or labelling classifier is entirely determined by the parsing configuration and depends on the type of the next action. The input to both classifiers is the concatenation of contextualized vectors extracted from a list of positions in the parsing configuration and specified as a list of feature templates.

In the ML-GAP transition system, a parsing configuration is defined by 3 data structures: a stack s containing subtrees, a double-ended queue d also containing subtrees, and a buffer b containing the yet unprocessed tokens. We use the following 11 templates:[7]

- the left-most and right-most token of first and second element in s and d (8 templates in total: $s_0.l$, $s_0.r$, $s_1.l$, $s_1.r$, $d_0.l$, $d_0.r$, $d_1.l$, $d_1.r$);[8]
- the next token in the buffer (b_0);
- the contextualized embeddings corresponding to the start of sentence and end of sentence symbols.

	P	R	F	DP	DR	DF	POS
Dev corpus							
ML-GAP	92.0	91.9	92.0	82.2	70.4	75.9	97.3
ML-GAP+FT	92.7	92.7	92.7	84.2	72.9	78.1	97.4
ML-GAP+BERT	95.0	95.1	95.0	86.2	85.4	85.8	97.6
Test corpus							
ML-GAP	91.8	91.0	91.4	82.6	67.7	74.4	97.6
ML-GAP+FT	92.5	92.1	92.3	85.0	69.6	76.5	97.7
ML-GAP+BERT	95.2	94.8	95.0	85.3	79.9	82.5	97.9

Table 2: Detailed results on the development and test sets of the DPTB. P and R are precision and recall; DP, DR, DF are the discontinuous precision, recall and F-score.

Overall results We report overall development and test results in Table 1 (see Appendix A for details about training), and compare them to published results on the DPTB dataset. For a more comprehensive evaluation of the parser, including results on the Tiger (Brants et al., 2002)[9] and Negra (Skut et al., 1997) German corpora, we refer the reader to Table 7 of Appendix B.

The ML-GAP setting improves over Coavoux et al. (2019) by 0.4 and 3.1 respectively for the F and DF metrics, which we attribute to the hyperparameter search. In both the supervised setting and the 'pretrained embeddings' setting, our parser's results lag behind Corro (2020), the current state of the art. However it is noticeably more accurate on discontinuous constituents (more than 10 absolute DF difference).

In the BERT-finetuning setting, the F measure of the ML-GAP+BERT model slightly outperforms the span-based parser of Corro (2020). The use

[7] This is an extension over the set of 7 templates from Coavoux et al. (2019).

[8] We use s_i (resp. d_i) to address the $i-1^{\text{th}}$ subtree of s (resp. d), and $.l/.r$ to address the left-most or right-most token yielded by the subtree.

[9] We use the SPMRL split (Seddah et al., 2013).

		Labelled				
Phenomenon	Count	Exact match	Partial match	Precision	Recall	F_1
Extraction	91	85.7 (+13.2, +16.5)	92.3 (+7.7, +9.9)	91.6 (+5.6, +7.1)	88.1 (+11.2, +13.1)	89.8 (+8.6, +10.3)
Fronted quotation	71	95.8 (+2.8, +4.3)	95.8 (+1.4, +2.8)	95.8 (+1.4, +4.1)	95.8 (+1.4, +2.8)	95.8 (+1.4, +3.5)
Discontinuous dependency	37	64.9 (+37.9, +46.0)	70.3 (+40.6, +46.0)	84.4 (-7.3, -5.6)	61.4 (+36.4, +40.9)	71.1 (+31.8, +37.8)
Circumpositioned quotation	16	81.2 (+18.7, +18.7)	100.0 (+12.5, +6.2)	94.1 (-1.0, +1.1)	96.0 (+18.0, +16.0)	95.0 (+9.3, +9.0)
It-extraposition	12	75.0 (+0.0, +33.3)	75.0 (+0.0, +33.3)	90.0 (+0.0, +6.7)	75.0 (+0.0, +33.3)	81.8 (+0.0, +26.2)
Extraction+fronted quotation	7	100.0 (+0.0, +14.3)	100.0 (+0.0, +0.0)	100.0 (+0.0, +0.0)	100.0 (+0.0, +5.3)	100.0 (+0.0, +2.7)
Discontinuous dependency+extraction	5	60.0 (+40.0, +60.0)	100.0 (+20.0, +0.0)	88.2 (+9.6, +16.8)	88.2 (+23.5, +29.4)	88.2 (+17.2, +23.7)
Extraction+extraction	5	80.0 (+20.0, +20.0)	100.0 (+0.0, +0.0)	94.4 (-5.6, -5.6)	100.0 (+23.5, +23.5)	97.1 (+10.4, +10.4)
Subject inversion	5	100.0 (+40.0, +40.0)	100.0 (+40.0, +40.0)	100.0 (+0.0, +0.0)	100.0 (+40.0, +40.0)	100.0 (+25.0, +25.0)
		Unlabelled				
Phenomenon	Count	Exact match	Partial match	Precision	Recall	F_1
Extraction	91	85.7 (+12.1, +15.4)	92.3 (+7.7, +9.9)	92.4 (+5.5, +7.7)	89.3 (+10.0, +12.0)	90.8 (+7.9, +10.0)
Fronted quotation	71	95.8 (+2.8, +4.3)	95.8 (+1.4, +2.8)	95.8 (+1.4, +4.1)	95.8 (+1.4, +2.8)	95.8 (+1.4, +3.5)
Discontinuous dependency	37	64.9 (+35.2, +43.3)	70.3 (+37.9, +43.3)	83.9 (-16.1, -16.1)	60.5 (+32.6, +37.2)	70.3 (+26.7, +32.6)
Circumpositioned quotation	16	100.0 (+25.0, +25.0)	100.0 (+12.5, +6.2)	100.0 (+0.0, +3.1)	100.0 (+21.1, +18.4)	100.0 (+11.8, +11.4)
It-extraposition	12	75.0 (+0.0, +33.3)	75.0 (+0.0, +33.3)	90.0 (+0.0, +6.7)	75.0 (+0.0, +33.3)	81.8 (+0.0, +26.2)
Extraction+fronted quotation	7	100.0 (+0.0, +14.3)	100.0 (+0.0, +0.0)	100.0 (+0.0, +0.0)	100.0 (+0.0, +5.6)	100.0 (+0.0, +2.9)
Discontinuous dependency+extraction	5	60.0 (+40.0, +60.0)	100.0 (+20.0, +0.0)	92.9 (+17.9, +17.9)	86.7 (+26.7, +26.7)	89.7 (+23.0, +23.0)
Extraction+extraction	5	80.0 (+20.0, +20.0)	100.0 (+0.0, +0.0)	94.1 (-5.9, -5.9)	100.0 (+18.8, +18.8)	97.0 (+7.3, +7.3)
Subject inversion	5	100.0 (+40.0, +40.0)	100.0 (+40.0, +40.0)	100.0 (+0.0, +0.0)	100.0 (+40.0, +40.0)	100.0 (+25.0, +25.0)

Table 3: Per-phenomenon results on the test suite for the ML-GAP+BERT model. Its absolute improvements over respectively the ML-GAP+FT and the ML-GAP model are in parentheses. We handle separately sentences that have several occurrences of phenomena resulting in discontinuities (+ symbol), we exclude combinations with fewer than 5 occurrences.

of BERT seems to cancel the benefits of the exact decoding permitted by the span-based approach of Corro (2020). On the DF metric, the gap is even larger (13.6 absolute difference). We attribute this difference to the fact that Corro (2020)'s parser is restricted to a certain type of discontinuities and cannot construct certain trees. Moreover, the transition-based paradigm enables a parser to use more fine-grained features than span-based parsers, which is particularly helpful for predicting discontinuous constituents.

6 Results and Discussion

In this section, we focus on the comparisons of our 3 models to assess the contribution of BERT to discontinuous parsing. We first focus on English, using the two resources we introduced in Sections 3 and 4. Then we provide and discuss results on German, using `discosuite` (Maier et al., 2014).

6.1 English

The improvements brought by BERT may come from its syntactic knowledge. However, they might also be a result of its extended lexical knowledge (providing more lexical information about out-of-vocabulary or rare words that might be known but do not take part in discontinuous structures in the training set). The ML-GAP+FT model provides a control setting, where the 'static' pretrained embeddings provide additional lexical information.

Overall effect of pretraining We provide detailed results (precision, recall, F) in Table 2. It had been reported that discontinuous parsers often have a large gap between precision (higher) and recall (lower) on discontinuities on both German and English corpora (Maier, 2015; Coavoux and Cohen, 2019; Stanojević and Steedman, 2020; Corro, 2020). The use of BERT tends to fill this gap, with a much stronger effect on recall (+15 DR on development set over **ml-gap**) than on precision (+4 DP). BERT leads the parser to better detect syntactic discontinuities compared to a supervised model (ML-GAP).

On the contrary, ML-GAP+FT provides only a small improvement over ML-GAP (+2.2 dev DF), which is split almost equally between precision (+2.0 DP) and recall (+2.5 DR). The striking difference between ML-GAP+FT and ML-GAP+BERT strongly suggests that BERT's contribution cannot be reduced to its extended lexical knowledge.

Per-phenomenon evaluation We report results on the test suite in Table 3, in the labelled case (upper part) and the unlabelled case (lower part). For each metric, we report the result of the ML-GAP+BERT model, as well as its absolute difference with, respectively the ML-GAP+FT and the ML-GAP model.

First, when comparing ML-GAP+BERT and ML-GAP, we observe a large improvement on all phenomena and almost all metrics. When comparing

Error type	ML-GAP+BERT Count	ML-GAP+BERT Nodes	ML-GAP+FT Count	ML-GAP+FT Nodes	ML-GAP Count	ML-GAP Nodes
PP Attachment	275 (-40.6%)	609 (-39.8%)	440 (-5.0%)	980 (-3.2%)	463	1012
⌞ discontinuous	17 (-34.6%)	27 (-34.1%)	26 (+0.0%)	40 (-2.4%)	26	41
Single Word Phrase	257 (-25.7%)	318 (-22.6%)	310 (-10.4%)	376 (-8.5%)	346	411
Unclassified	200 (-47.2%)	263 (-48.8%)	318 (-16.1%)	427 (-16.9%)	379	514
NP Internal Structure	191 (-26.8%)	221 (-32.4%)	218 (-16.5%)	283 (-13.5%)	261	327
⌞ discontinuous	1 (+∞%)	1 (+∞%)	1 (+∞%)	2 (+∞%)	0	0
Modifier Attachment	186 (-23.8%)	344 (-24.4%)	221 (-9.4%)	408 (-10.3%)	244	455
⌞ discontinuous	9 (-30.8%)	13 (-27.8%)	10 (-23.1%)	15 (-16.7%)	13	18
Unary	185 (-41.1%)	185 (-41.1%)	271 (-13.7%)	271 (-13.7%)	314	314
⌞ discontinuous	2 (+0.0%)	2 (+0.0%)	2 (+0.0%)	2 (+0.0%)	2	2
Different label	166 (-22.4%)	335 (-22.1%)	211 (-1.4%)	426 (-0.9%)	214	430
⌞ discontinuous	6 (+0.0%)	15 (+7.1%)	6 (+0.0%)	16 (+14.3%)	6	14
Clause Attachment	116 (-37.6%)	292 (-37.7%)	170 (-8.6%)	450 (-4.1%)	186	469
⌞ discontinuous	16 (-40.7%)	25 (-34.2%)	31 (+14.8%)	44 (+15.8%)	27	38
Co-ordination	84 (-45.8%)	210 (-47.2%)	127 (-18.1%)	312 (-21.6%)	155	398
⌞ discontinuous	3 (-25.0%)	8 (+14.3%)	4 (+0.0%)	8 (+14.3%)	4	7
NP Attachment	51 (-51.9%)	168 (-43.6%)	94 (-11.3%)	312 (+4.7%)	106	298
⌞ discontinuous	9 (-55.0%)	19 (-20.8%)	14 (-30.0%)	18 (-25.0%)	20	24
VP Attachment	19 (-63.5%)	62 (-67.7%)	40 (-23.1%)	135 (-29.7%)	52	192
⌞ discontinuous	2 (-71.4%)	2 (-86.7%)	7 (+0.0%)	13 (-13.3%)	7	15
XoverX Unary	10 (+0.0%)	10 (+0.0%)	10 (+0.0%)	10 (+0.0%)	10	10

Table 4: Error types for both models (development set). Absolute differences with the ML-GAP model are given in parentheses.

labelled and unlabelled results, we observe very small differences (< 1) except for the case of circumpositioned quotations. This is due to some cases where the discontinuous quotation phrase has an unfrequent label (FRAG or SINV). Overall, subject inversions,[10] fronted quotations and circumpositioned quotations are almost perfectly detected by the ML-GAP+BERT system with DF scores over 95, and high exact match (at least in the unlabelled case for circumpositioning). On the other hand, discontinuous dependencies and *it*-extrapositions, and to a lower extent extractions, have DF scores below 90, despite the huge effect of BERT (respectively +35.2 and +25 absolute improvement on exact match for discontinuous dependencies and *it*-extrapositions).

Secondly, the improvement brought by fast-text embeddings is consistently very small (around +2F), except on 2 types of phenomena: *it*-extrapositions (where BERT does not improve over fast-text), and to a lower extent discontinuous dependencies (+6F for fast-text, +37.8F for BERT). This result suggests that the difficulty to parse these phenomena stemmed, at least partly from a lack of lexical knowledge.

Error Analysis We report results of the error type classifier for both models in Table 4. For each error type, we report (i) the overall count of occurrences, (ii) the number of occurrences where the correction involved a discontinuous node among them, and (iii) the total number of nodes involved (a single error can cause multiple wrong nodes), as done by Kummerfeld et al. (2012).

[10]Note that we have a small sample for this phenomenon (5 instances).

Overall, we observe an important decrease across all types of errors, with error reduction rates often close to 40% (e.g. 45% fewer occurrences of PP attachment errors) for ML-GAP+BERT and around 20% for ML-GAP+FT.

The picture is slightly different if we look at errors involving discontinuous constituents. Indeed, the use of BERT drastically reduces the main sources of errors (PP/VP/NP attachment, modifier attachment, coordination), while having no effect on other types of structure (NP internal structure, unary constituent, label). In contrast, fast-text only improves modifier and NP attachments, and even introduces clause attachment errors.

6.2 German

In order to provide additional context to our results on English, we further experiment with the same parsing models on German, using the test-suite built by Maier et al. (2014) on the German Tiger corpus. They constructed this test-suite by first identifying and classifying discontinuous phenomena in the 1500 first sentences of the Tiger corpus; and then they selected 15[11] sentences for each identified phenomenon. In total, `discosuite` contains 180 occurrences across 151 sentences. Each occurrence corresponds to a single discontinuous constituent.

We train our parsers on a modified version of the SPMRL Tiger split, where the 151 sentences are removed from the training set. We then parse the 151 sentences and use the labelled and unlabelled recall on target constituents to evaluate the corresponding phenomena. We provide results on the testsuite in Table 5, using the same settings as in English (ML-GAP, ML-GAP+FT, ML-GAP+BERT). We refer the reader to Maier et al. (2014) for the descriptions of specific phenomena. To the best of our knowledge, no prior parsing work used this testsuite for evaluation since its release.

Due to a finer-grain classification, there are only few instances for each type. Hence, we only comment on general patterns. Overall, fast-text provides small improvement on 6 types of phenomena (over 14). In constrast, BERT improves on every type of phenomenon, with largest increases for extrapositions of an element of a coordination, extrapositions involving a focus adverb (eg. adverb in the main clause modifying a subordinate clause), and local movement (involves discontinuities that

[11]or fewer for rarer phenomena.

Phenomenon	Occurrences	Labelled recall			Unlabelled recall		
		ML-GAP+BERT (Δ)	ML-GAP+FT (Δ)	ML-GAP	ML-GAP+BERT (Δ)	ML-GAP+FT (Δ)	ML-GAP
Extraposed arguments	15	86.7 (+27)	66.7 (+7)	60	86.7 (+20)	66.7 (=)	66.7
Extraposed modifiers	15	73.3 (+13)	73.3 (+13)	60	73.3 (+7)	73.3 (+7)	66.7
Extraposition (comparison)	15	73.3 (+7)	66.7 (=)	66.7	80 (+7)	73.3 (=)	73.3
Extraposition (coordination)	15	40 (+33)	33.3 (+27)	6.7	60 (+47)	40 (+27)	13.3
Extraposition (focus adverb)	5	80 (+40)	40 (=)	40	80 (+20)	60 (=)	60
Local Movement (clause)	10	30 (+30)	0 (=)	0	30 (+30)	0 (=)	0
Local Movement (phrase)	10	40 (+30)	10 (=)	10	50 (+40)	10 (=)	10
Placeholder/repeated element	15	93.3 (+7)	80 (-7)	86.7	93.3 (+7)	80 (-7)	86.7
Parentheticals	15	86.7 (+13)	86.7 (+13)	73.3	93.3 (+7)	93.3 (+7)	86.7
Pronouns	15	73.3 (+13)	66.7 (+7)	60	73.3 (+13)	66.7 (+7)	60
Scrambling	15	60 (+13)	80 (+33)	46.7	60 (+13)	80 (+33)	46.7
Topicalization Other	10	10 (+10)	0 (=)	0	10 (+10)	10 (+10)	0
Topicalization VP HD	10	80 (+10)	50 (-20)	70	80 (+10)	50 (-20)	70
Topicalization VP mod/arg	15	93.3 (+20)	73.3 (=)	73.3	93.3 (+20)	73.3 (=)	73.3

Table 5: Results of ML-GAP+BERT, ML-GAP+FT and ML-GAP on `discosuite` (Maier et al., 2014). Absolute difference with ML-GAP model is indicated in parentheses.

do not cross clause boundaries). These are also the most difficult phenomena to predict correctly (< 50 recall).

We observe there is not a large difference between labelled and unlabelled scores, suggesting that finding the correct structure is the main difficulty.

7 Conclusion

We introduced two resources for fine-grained automatic error analysis of English discontinuous constituency parsers. First, we construct and release a test-suite for the range of syntactic phenomena responsible for the discontinuous structures in the discontinuous version of the Penn Treebank. Second, we extend the Berkeley parser analyser to the analysis of discontinuous constituency trees. We apply these resources to study the contribution of BERT to discontinuous parsing of English.

Overall, on almost all phenomena, BERT brings an improvement over a fast-text baseline. We found that BERT leads to almost perfect detection for some phenomena (subject inversion, fronted quotations, circumpositioned quotations). Moreover, there is still a wide room for improvement for extractions (despite the high frequency of this type of structures in the corpus), *it*-extrapositions, and discontinuous dependencies. In future work, we plan to address these limitations with targeted data-augmentation methods. We also plan to evaluate other pretrained language models to assess whether they exhibit the same error patterns as BERT.

Acknowledgments

I thank Sacha Beniamine, Caio Corro, Didier Schwab and 2 sets of 3 reviewers for feedback on previous versions of this paper. All trees are drawn with `discodop`.

References

Lei Jimmy Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. 2016. Layer normalization. *CoRR*, abs/1607.06450.

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 Brants, Stefanie Dipper, Silvia Hansen, Wolfgang Lezius, and George Smith. 2002. The tiger treebank. In *Proceedings of the workshop on treebanks and linguistic theories*, volume 168.

Maximin Coavoux and Shay B. Cohen. 2019. Discontinuous constituency parsing with a stack-free transition system and a dynamic oracle. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 204–217, Minneapolis, Minnesota. Association for Computational Linguistics.

Maximin Coavoux and Benoît Crabbé. 2017. Incremental discontinuous phrase structure parsing with the GAP transition. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 1259–1270, Valencia, Spain. Association for Computational Linguistics.

Maximin Coavoux, Benoît Crabbé, and Shay B. Cohen. 2019. Unlexicalized transition-based discontinuous constituency parsing. *Transactions of the Association for Computational Linguistics*, 7:73–89.

Caio Corro. 2020. Span-based discontinuous constituency parsing: a family of exact chart-based algorithms with time complexities from O(nˆ6) down to O(nˆ3). In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2753–2764, Online. Association for Computational Linguistics.

Caio Corro, Joseph Le Roux, and Mathieu Lacroix. 2017. Efficient discontinuous phrase-structure parsing via the generalized maximum spanning arborescence. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1644–1654, Copenhagen, Denmark. Association for Computational Linguistics.

Andreas van Cranenburgh and Rens Bod. 2013. Discontinuous parsing with an efficient and accurate DOP model. In *Proceedings of the 13th International Conference on Parsing Technologies (IWPT 2013)*, pages 7–16, Nara, Japan. Assocation for Computational Linguistics.

Andreas van Cranenburgh, Remko Scha, and Rens Bod. 2016. Data-oriented parsing with discontinuous constituents and function tags. *Journal of Language Modelling*, 4(1):57–111.

James Cross and Liang Huang. 2016. Span-based constituency parsing with a structure-label system and provably optimal dynamic oracles. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1–11, Austin, Texas. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Kilian Evang. 2011. Parsing discontinuous constituents in English. Master's thesis, University of Tübingen.

Kilian Evang and Laura Kallmeyer. 2011. PLCFRS parsing of English discontinuous constituents. In *Proceedings of the 12th International Conference on Parsing Technologies*, pages 104–116, Dublin, Ireland. Association for Computational Linguistics.

Daniel Fernández-González and Carlos Gómez-Rodríguez. 2020. Discontinuous constituent parsing with pointer networks. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 7724–7731. AAAI Press.

Daniel Fernández-González and Carlos Gómez-Rodríguez. 2020. Multitask pointer network for multi-representational parsing. *arXiv preprint arXiv:2009.09730*.

Daniel Fernández-González and André F. T. Martins. 2015. Parsing as reduction. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1523–1533, Beijing, China. Association for Computational Linguistics.

Ryan Gabbard, Seth Kulick, and Mitchell Marcus. 2006. Fully parsing the Penn Treebank. In *Proceedings of the Human Language Technology Conference of the NAACL, Main Conference*, pages 184–191, New York City, USA. Association for Computational Linguistics.

Kilian Gebhardt. 2018. Generic refinement of expressive grammar formalisms with an application to discontinuous constituent parsing. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 3049–3063, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Kilian Gebhardt. 2020. Advances in using grammars with latent annotations for discontinuous parsing. In *Proceedings of the 16th International Conference on Parsing Technologies and the IWPT 2020 Shared Task on Parsing into Enhanced Universal Dependencies*, pages 91–97, Online. Association for Computational Linguistics.

Edouard Grave, Piotr Bojanowski, Prakhar Gupta, Armand Joulin, and Tomas Mikolov. 2018. Learning word vectors for 157 languages. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan. European Language Resources Association (ELRA).

Kristina Gulordava, Piotr Bojanowski, Edouard Grave, Tal Linzen, and Marco Baroni. 2018. Colorless green recurrent networks dream hierarchically. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1195–1205, New Orleans, Louisiana. Association for Computational Linguistics.

Jennifer Hu, Jon Gauthier, Peng Qian, Ethan Wilcox, and Roger Levy. 2020. A systematic assessment of syntactic generalization in neural language models. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1725–1744, Online. Association for Computational Linguistics.

Laura Kallmeyer. 2010. *Parsing Beyond Context-Free Grammars*, 1st edition. Springer Publishing Company, Incorporated.

Diederik P Kingma and Jimmy Lei Ba. 2015. Adam: A method for stochastic gradient descent. In *ICLR: International Conference on Learning Representations*, pages 1–15.

Nikita Kitaev, Steven Cao, and Dan Klein. 2019. Multilingual constituency parsing with self-attention and pre-training. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3499–3505, Florence, Italy. Association for Computational Linguistics.

Nikita Kitaev and Dan Klein. 2018. Constituency parsing with a self-attentive encoder. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2676–2686, Melbourne, Australia. Association for Computational Linguistics.

Sandra Kübler, Ines Rehbein, and Josef van Genabith. 2009. A testsuite for testing parser performance on complex german grammatical constructions. In *Proceedings of the Seventh International Workshop on Treebanks and Linguistic Theories : Groningen, Netherlands, January 23-24, 2009*. MP.

Jonathan K. Kummerfeld, David Hall, James R. Curran, and Dan Klein. 2012. Parser showdown at the Wall Street corral: An empirical investigation of error types in parser output. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1048–1059, Jeju Island, Korea. Association for Computational Linguistics.

Jonathan K. Kummerfeld and Dan Klein. 2017. Parsing with traces: An O(n4) algorithm and a structural representation. *Transactions of the Association for Computational Linguistics*, 5:441–454.

Jonathan K. Kummerfeld, Daniel Tse, James R. Curran, and Dan Klein. 2013. An empirical examination of challenges in Chinese parsing. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 98–103, Sofia, Bulgaria. Association for Computational Linguistics.

Adhiguna Kuncoro, Chris Dyer, John Hale, Dani Yogatama, Stephen Clark, and Phil Blunsom. 2018. LSTMs can learn syntax-sensitive dependencies well, but modeling structure makes them better. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1426–1436, Melbourne, Australia. Association for Computational Linguistics.

Tal Linzen, Emmanuel Dupoux, and Yoav Goldberg. 2016. Assessing the ability of LSTMs to learn syntax-sensitive dependencies. *Transactions of the Association for Computational Linguistics*, 4:521–535.

Wolfgang Maier. 2015. Discontinuous incremental shift-reduce parsing. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1202–1212, Beijing, China. Association for Computational Linguistics.

Wolfgang Maier, Miriam Kaeshammer, Peter Baumann, and Sandra Kübler. 2014. Discosuite - a parser test suite for German discontinuous structures. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, pages 2905–2912, Reykjavik, Iceland. European Language Resources Association (ELRA).

Wolfgang Maier and Timm Lichte. 2016. Discontinuous parsing with continuous trees. In *Proceedings of the Workshop on Discontinuous Structures in Natural Language Processing*, pages 47–57, San Diego, California. Association for Computational Linguistics.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*, 19(2):313–330.

Rebecca Marvin and Tal Linzen. 2018. Targeted syntactic evaluation of language models. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1192–1202, Brussels, Belgium. Association for Computational Linguistics.

Tomas Mikolov, Edouard Grave, Piotr Bojanowski, Christian Puhrsch, and Armand Joulin. 2018a. Advances in pre-training distributed word representations. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan. European Language Resources Association (ELRA).

Tomas Mikolov, Edouard Grave, Piotr Bojanowski, Christian Puhrsch, and Armand Joulin. 2018b. Advances in pre-training distributed word representations. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan. European Language Resources Association (ELRA).

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 8024–8035.

Adwait Ratnaparkhi, Jeff Reynar, and Salim Roukos. 1994. A maximum entropy model for prepositional phrase attachment. In *Human Language Technology: Proceedings of a Workshop held at Plainsboro, New Jersey, March 8-11, 1994*.

Thomas Ruprecht and Richard Mörbitz. 2021. Supertagging-based parsing with linear context-free rewriting systems. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2923–2935, Online. Association for Computational Linguistics.

Djamé Seddah, Reut Tsarfaty, Sandra Kübler, Marie Candito, Jinho D. Choi, Richárd Farkas, Jennifer Foster, Iakes Goenaga, Koldo Gojenola Galletebeitia, Yoav Goldberg, Spence Green, Nizar Habash, Marco Kuhlmann, Wolfgang Maier, Joakim Nivre, Adam Przepiórkowski, Ryan Roth, Wolfgang Seeker, Yannick Versley, Veronika Vincze, Marcin Woliński, Alina Wróblewska, and Eric Villemonte de la Clergerie. 2013. Overview of the SPMRL 2013 shared task: A cross-framework evaluation of parsing morphologically rich languages. In *Proceedings of the Fourth Workshop on Statistical Parsing of Morphologically-Rich Languages*, pages 146–182, Seattle, Washington, USA. Association for Computational Linguistics.

Hiroyuki Seki, Takashi Matsumura, Mamoru Fujii, and Tadao Kasami. 1991. On multiple context-free grammars. *Theoretical Computer Science*, 88(2):191 – 229.

Wojciech Skut, Brigitte Krenn, Thorsten Brants, and Hans Uszkoreit. 1997. An annotation scheme for free word order languages. In *Fifth Conference on Applied Natural Language Processing*, pages 88–95, Washington, DC, USA. Association for Computational Linguistics.

Miloš Stanojević and Raquel G. Alhama. 2017. Neural discontinuous constituency parsing. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1666–1676, Copenhagen, Denmark. Association for Computational Linguistics.

Miloš Stanojević and Mark Steedman. 2020. Span-based LCFRS-2 parsing. In *Proceedings of the 16th International Conference on Parsing Technologies and the IWPT 2020 Shared Task on Parsing into Enhanced Universal Dependencies*, pages 111–121, Online. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 5998–6008.

Yannick Versley. 2016. Discontinuity (re)^2-visited: A minimalist approach to pseudoprojective constituent parsing. In *Proceedings of the Workshop on Discontinuous Structures in Natural Language Processing*, pages 58–69, San Diego, California. Association for Computational Linguistics.

K. Vijay-Shanker, David J. Weir, and Aravind K. Joshi. 1987. Characterizing structural descriptions produced by various grammatical formalisms. In *25th Annual Meeting of the Association for Computational Linguistics*, pages 104–111, Stanford, California, USA. Association for Computational Linguistics.

David Vilares and Carlos Gómez-Rodríguez. 2020. Discontinuous constituent parsing as sequence labeling. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2771–2785, Online. Association for Computational Linguistics.

Ethan Wilcox, Peng Qian, Richard Futrell, Miguel Ballesteros, and Roger Levy. 2019. Structural supervision improves learning of non-local grammatical dependencies. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 3302–3312, Minneapolis, Minnesota. Association for Computational Linguistics.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

A Training Details

The parser uses the pytorch library (Paszke et al., 2019) and the `transformers` interface (Wolf et al., 2020) to fine-tune BERT. For all 3 models (ML-GAP, ML-GAP+FT, ML-GAP+BERT), we only tuned the batch size and the learning rate. We report all hyperparameters and final configurations in Table 6 of Appendix A. We used the Adam optimizer (Kingma and Ba, 2015), divide learning rate by 2 when the model shows no improvement on the dev set for 5 epochs and stop training after 3 such cycles with no improvement. Finally, we maintain the average of parameters across iterations during training, and use the averaged parameters (instead of the final parameters) for the final model. We trained the parsers on gold sequences of configurations (teacher-forcing). We use a deterministic

Parser	ML-GAP/ML-GAP+FT	ML-GAP+BERT
Sentence encoder hyperparameters		
Character embedding	64	64
Character bi-LSTM state	256	128
Character dropout	0.2	0.2
Word embeddings	300	-
Word dropout	0.2/0	-
Sentence encoder	bi-LSTM	bi-LSTM
Sentence encoder state	400	400
Stacked bi-LSTM	2	2
Action scorer hyperparameters		
Activation	tanh	tanh
Hidden layers	1	1
Hidden size	250	250
Dropout	0.3	0.3
Optimization hyperparameters		
DPTB: batch size	$\{4, \mathbf{8}\}/\{\mathbf{4}, 8\}$	{16, **32**}
DPTB: learning rate	{0.001, **0.0015**}	{**0.00006**, 0.00008, 0.0001, 0.00012}
Tiger: batch size	$\{\mathbf{4}, 8\}/\{4, \mathbf{8}\}$	{16, **32**}
Tiger: learning rate	{0.001, **0.0015**}	{**0.00006**, 0.00008, 0.0001, 0.00012}
Negra: batch size	$\{\mathbf{4}, 8\}/\{\mathbf{4}, 8\}$	{16, **32**}
Negra: learning rate	{**0.001**, 0.0015}/{0.001, **0.0015**}	{**0.00006**, 0.00008, 0.0001, 0.00012}
DPTB: bert model	-	`bert-base-cased`
Negra, Tiger: bert model	-	`bert-base-german-cased`

Table 6: Hyperparameter search for all models (best configuration in bold). The '/' symbol indicates different final values for ML-GAP and ML-GAP+FT models.

oracle that prioritizes merges over shifts when both are possible, this implicitly corresponds to a left-binarization of n-ary constituents. Finally, we use greedy search for finding the best sequence of actions.

B Results on German Treebanks

System	Method type	DPTB		Tiger		Negra	
		F	DF	F	DF	F	DF
Fully supervised							
van Cranenburgh and Bod (2013), ≤ 40	GB	85.6		75.3$^{\neq}$		74.8	
Fernández-González and Martins (2015)	DB			77.3		77.0	
van Cranenburgh et al. (2016), ≤ 40	GB	87.0		78.2$^{\neq}$		76.8	
Maier (2015), gold POS	TB			74.7	18.8	77.0	19.8
Versley (2016)	GB			79.5			
Maier and Lichte (2016)	TB			76.5	16.3		
Corro et al. (2017)	DB	89.2					
Coavoux and Crabbé (2017)	TB			79.3			
Gebhardt (2018)	GB			75.1			
Coavoux and Cohen (2019)	TB	90.9	67.3	82.5	55.9	83.2	56.3
Coavoux et al. (2019)	TB	91.0	71.3	82.7	55.9	83.2	54.6
Fernández-González and Gómez-Rodríguez (2020)	DB			84.6	57.9	83.7	54.7
Stanojević and Steedman (2020)	CB	90.5	67.1	83.4	53.5	83.6	50.7
Corro (2020)	CB	**92.7**	64.2	85.5	53.8	86.2	54.1
Ruprecht and Mörbitz (2021)	GB	90.1	72.9	82.5	55.9	82.7	49.0
Fernández-González and Gómez-Rodríguez (2020)	DB			**86.6**	**62.6**	**86.8**	**69.5**
Gebhardt (2020)	GB			77.7	40.7	81.7	43.5
This work: ML-GAP	TB	91.4	**74.4**	82.9	57.4	82.3	55.6
Semi-supervised (Pretrained embeddings)							
Stanojević and Alhama (2017)	TB			77.0			
Corro (2020)	CB	**92.9**	64.9	85.2	51.2	86.3	56.1
Fernández-González and Gómez-Rodríguez (2020)	DB			**85.7**	60.4	**85.7**	58.6
Ruprecht and Mörbitz (2021)	GB	91.8	76.1	85.1	61.0	86.5	61.9
This work: ML-GAP+FT	TB	92.3	**76.5**	85.2	**61.1**	85.6	**60.9**
Semi-supervised (Bert-base)							
Corro (2020)	CB	94.8	68.9	90.0	62.1	91.6	66.1
Fernández-González and Gómez-Rodríguez (2020)	DB			89.8	71.0	91.0	**76.6**
Vilares and Gómez-Rodríguez (2020)	SL	91.9	50.8	84.6	51.1	83.9	45.6
Ruprecht and Mörbitz (2021)	GB	93.3	80.5	88.3	69.0	90.9	72.6
This work: ML-GAP+BERT	TB	**95.0**	**82.5**	**90.2**	**72.9**	**91.7**	73.3

Table 7: Results on the DPTB, Tiger and Negra corpora (test sets). DF: discontinuous F-score. Methods: GB: grammar-based, TB: transition-based, CB: grammarless chart-based, SL: sequence-labelling based, DB: dependency-conversion based. $\neq$: different train/dev/test split.