What's in a Span? Evaluating the Creativity of a Span-Based Neural Constituency Parser

Daniel Dakota
Uppsala University
daniel.dakota@lingfil.uu.se

Sandra Kübler
Indiana University
skuebler@indiana.edu

Abstract

Constituency parsing is generally evaluated superficially, particularly in a multiple language setting, with only F-scores being reported. As new state-of-the-art chart-based parsers have resulted in a transition from traditional PCFG-based grammars to span-based approaches (Stern et al., 2017; Gaddy et al., 2018), we do not have a good understanding of how such fundamentally different approaches interact with various treebanks as results show improvements across treebanks (Kitaev and Klein, 2018), but it is unclear what influence annotation schemes have on various treebank performance (Kitaev et al., 2019). In particular, a span-based parser's capability of creating novel rules is an unknown factor. We perform an analysis of how span-based parsing performs across 11 treebanks in order to examine the overall behavior of this parsing approach and the effect of the treebanks' specific annotations on results. We find that the parser tends to prefer flatter trees, but the approach works well because it is robust enough to adapt to differences in annotation schemes across treebanks and languages.

1 Introduction

Constituency parsing has slowly shifted from the once dominant PCFG-based grammar approaches to span-based neural approaches with a high degree of success (Hall et al., 2014; Cross and Huang, 2016; Gaddy et al., 2018; Kitaev and Klein, 2018; Kitaev et al., 2019). Such a parser is not restricted to the grammar extracted from training data anymore, but instead learns which constituents to group and which new constituent label to assign from its current configuration. This is parallel to data-driven dependency parsing (Kübler et al., 2009) in that the parser learns parsing actions rather than probabilities of rules; and it has been used in shift-reduce constituency parsing (Cross and Huang, 2016)

Not being limited by the grammar rules from the training data is one reason for the success of span-based neural parsers. Being able to create novel rules[1] solves one of the problems of grammar-based parsers since it is often the case that test sentences contain rules in their gold standard tree that are not present in the training data, which limited a grammar-based parser's ability to reach perfect precision and recall.

In a multilingual setting, where we parse more than one language without adjusting the parsing approach to individual languages, there is an interesting interaction between the parser's ability to create new rules and the language and treebank in question. For example, the flatter the gold standard trees are, the more specific the rules, the more different rules are needed to cover all variants of a specific phrase, and the higher the chance that some of these rules will not be present in the training data. Since constituent treebanks were annotated with language dependent annotation schemes, there exist dramatic differences in terms of span length, or the number of constituent labels, depending on how much branching an annotation scheme requires (Seddah et al., 2013, 2014).

In a multilingual setting, the parser needs to be general enough to capture linguistic phenomena across languages. Thus, it is important to understand how the underlying mechanism interacts with the annotations. However, an in-depth analysis of constituency parses is difficult as trees can differ in many different aspects, and errors tend to propagate higher up in the tree. In order to better understand some of the behaviors of a span-based approach, we focus on understanding the effect of newly created

[1]We use the term 'rule' to refer to a traditional phrase structure rule, i.e., including the mother node and the sequence of daughter nodes; we use 'span' to refer to a sequence of daughters.

rules across a range of treebanks. More specifically, we investigate which novel rules the parser can create correctly, and which types of errors the parser makes when it creates incorrect rules.

We use the state-of-the-art Berkeley Neural Parser (Kitaev et al., 2019) on the SPMRL data set (Seddah et al., 2013, 2014), and we add the English Penn Treebank (Marcus et al., 1993) and the Penn Chinese Treebank (Xue et al., 2005) to cover typologically different languages.

We begin with brief summaries of span-based constituency parsing and in-depth evaluations in section 2 before describing our methodology in section 3. We then present an overview of treebank statistics in section 4, followed by an analysis of known rules in section 5, and of novel rules in section 6, before concluding in section 7.

2 Previous Work

2.1 Span-Based Neural Constituency Parsing

The shift from grammar-based parsing to data-driven parsing has first been implemented for transition-based parsing (Nivre, 2003) and was adapted for shift-reduce constituency parsing early on by Sagae and Lavie (2005). However, it has only recently been implemented in chart-based parsers (Stern et al., 2017; Gaddy et al., 2018). Traditional chart parsers based on PCFG grammars had one major limitation: They assumed a complete grammar, extracted from the training data. This worked well for languages with a low ratio of unique rules, where most of the rules were observed in the training data. However, for languages with annotation schemes preferring flat trees, this was not true. The first approach to address this limitation was (horizontal) Markovization (Collins, 1999; Klein and Manning, 2003), which allowed the parser a strictly limited type of creating new rules.

This pivotal advantages of current span-based neural parsers allows them more freedom in creating novel rules by decoupling decisions on spans from a strict set of possible spans, and also decoupling decisions on spans from their constituent labels (Cross and Huang, 2016). This freedom has been made possible empirically by the use of LSTMs (Hochreiter and Schmidhuber, 1997), which provide a rich context for making span decisions.

Span-based parsing encodes a constituency tree on the span level rather than the word level. While a span may only cover a single word, it can also represent a sequence of words (i.e., a span). The constituent label of a given span is often calculated independently of the span, resulting in a two-step constituent building process. A tree is considered to be the collection of spans, thus the goal of a span-based parser is to build up individual spans and constituent labels into a well-formed tree.

Current span-based neural parsers tend to use three aspects: word representations, span representations, and separate label scoring (Gaddy et al., 2018). Note that one of the inherent flexibilities of this approach is the separation of the identification of a span and its constituent label into two different, yet ultimately joined, scoring functions. Thus, the parser is able to balance choosing the type of a span against identifying the best split point of a larger span. This allows the parser to have a much more robust ability to generate novel rules. A tree is then decomposed into the scores of its constituents, with the optimal tree being determined via a modified version of the CYK algorithm. Span-based neural parsing has shown a clear improvement over prior approaches in a multilingual setting (Hall et al., 2014; Kitaev and Klein, 2018), and currently such approaches give state-of-the-art-performance across multiple treebanks (Kitaev et al., 2019).

2.2 In-depth Error Analysis

Performing a detailed analysis of constituency parses is a difficult task. Since parsing is performed incrementally, a parsing error often propagates up the tree, thus presenting a complex situation in which it is impossible to determine the original cause of an incorrect tree. The situation is even worse in a multilingual setting since there are significant differences between the annotation schemes for the individual languages. While the standard evaluation consists of reporting EVALB metrics (Sekine and Collins, 1997), there are investigations into single linguistic phenomena (e.g., discontinuous constituents (van Cranenburgh et al., 2016) or prepositional attachment (de Kok et al., 2017)) in order to isolate and analyze specific behaviors. This means there is often a lack of depth in understanding the various interactions between a parsing approach and different annotation schemes.

One of the earliest in-depth investigations was performed by Levy and Manning (2003) on the Penn Chinese Treebank. They found that many errors are due to annotation decisions or ambiguities. In other words, annotation decisions play an important role in understanding global parsing decisions. This is further emphasized by Kübler (2005) and Rehbein and van Genabith (2007) in their discussions on the comparability of performance and evaluation based on two German treebanks with different annotation schemes.

Kummerfeld et al. (2012, 2013) performed a set of analyses on the English PTB and Chinese PTB respectively by developing a visualization tool for errors.[2] This provided deeper insights into the behavior of various constituency parsers in order to identify systematic linguistic errors. Errors were mainly PP and clause attachment for English. Additionally, gold POS tags did not lead to an improvement in verb-argument attachment and coordination scope in Chinese. This is supported by Dakota (2018) who found that improvements on the POS level on the German TiGer treebank (Brants et al., 2004) did not necessarily equally percolate to higher level projections. Many performance aspects can be attributed to various treebank annotation decisions and are not easily overcome, particularly if the annotations are inconsistent.

Work by Nguyen et al. (2018) investigated the low resource language of Vietnamese by comparing several non-neural and neural parsers to identify sources of common errors. While POS tagging errors did contribute to problems for specific phrase types, they were not the predominant issue in regards to NPs and PPs, rather inherent ambiguity in the language was the main cause. In the same vein, work by Kim and Park (2020) for Korean shows that neural parsers improve substantially in error reduction over their non-neural counterparts, but the most common remaining errors are still those related to long distance attachments.

3 Methodology

Treebanks We use the SPMRL dataset (see (Seddah et al., 2013, 2014) for details on the treebanks and annotation schemes), the English Penn Treebank (Marcus et al., 1993), and the Penn Chinese Treebank V5 (Xue et al., 2005) as our constituency treebanks, the former being the de-facto standard for multilingual evaluation. We use the standard train, dev, test splits for each treebank[3]. The dev sets are used to optimize the parser, and we evaluate on the test sets. We use the 5k training sets. From the dev and test sets, we select the first 500 sentences in the dev and test sets to ensure replicability. The small data sizes were chosen for two reasons: 1) This normalizes the size of the data sets across the languages and gives us a better basis for cross-linguistic comparisons. 2) It maximizes the effect of rule creation since the parser has not been exposed to as many rules, providing us with more examples of possible new rules. We do, however, recognize that results would be different when using the full sets or any random sampling of the test sets, as shown by Dakota and Kübler (2017).

Parser We use the Berkeley Neural Parser, which has previously achieved state of the art results on all datasets used here (Kitaev et al., 2019). The parser uses a span-based model derived from Stern et al. (2017) and subsequently Gaddy et al. (2018) with self-attention and pre-training. We follow the experimental setup provided with the parser with one modification: Because of the small training set size, we check performance on the dev set once per epoch rather than the default of four. We also remove all grammatical functions (such as subject or direct object) from the data before training, as the parser, when given the full treebank representations, will use grammatical functions for optimization, but then discards them when evaluating. This would introduce another variable to our experiments.

Evaluation We evaluate using the scorer for the SPMRL 2014 shared task[4], a re-implementation of EVALB[5] that was distributed with the task as well as being the evaluation used for development of the parser.

The following analyses are performed per sentence rather than per rule. We focus on an analysis of the test set with regard to the training set. We recognize the influence of the dev set for parser optimization, but considering this interaction is beyond the scope of this work. Most of the results

[2]https://github.com/jkkummerfeld/berkeley-parser-analyser/

[3]We use standard train, dev, and test sets for English and Chinese, which equates to the the first 5000, 500, and 500 sentences in the train, dev, and test for both. For Arabic, we remove sentences longer than 266 words as these sentences are too long for BERT and thus not used by the Berkeley Neural Parser as per the experimental setup provided with the parser, resulting in 4994 training sentences. The dev set was not affected. For Chinese, 8 sentences were removed from train5k and 3 from dev due to lack of roots, resulting in 4992 and 347 sentences respectively. We also note that for both Swedish and Hebrew, 5000 sentences is the max training size available, and Swedish has only 494 dev sentences, Hebrew 500.

[4]http://spmrl.org/spmrl2014-sharedtask

[5]http://nlp.cs.nyu.edu/evalb/

on the treebanks we use were reported on test. We perform our analysis on the test sets to allow for a measure of comparability.

4 Treebank Rule Statistics

Before we can start looking at parser performance, we need a better understanding of the differences between treebanks. In Table 1 we present rule statistics and evaluation across the train5k and test500 sets. Given the nature of our data splits, there are no adequate direct comparisons that can be made with the performance, with the closest being Dakota and Kübler (2017) who examined the variation in performance evaluation when randomly selecting 500 test sentences on a subset of languages in the SPRML dataset, the English PTB, and Chinese PTB; however, they used a PCFG-LA parser (Petrov and Klein, 2007) and no optimization on a dev set was performed.

A first look at the numbers in Table 1 shows radically different numbers of rules present in the various sets, despite very similar set sizes. The same holds for the number of rule types (i.e., a rule `S` $\rightarrow$ `NP VP` would count as one rule type regardless of how often it occurs). Thus for Arabic, although there are almost 230 000 total rules in train, there are only roughly 4 200 rule types to which the parser is exposed to during training.

We also provide the ratio of rule types to total rules and number of constituent types (Const; e.g., NP or VP) per data set. This gives a general indication of the rule complexity inherent in each of the data sets, though it does not show the distribution within each of the constituent types[6]. We see that most languages in the train5k range between a ratio of 0.04 and 0.06, but there are some extreme outliers with Polish at 0.0043 and German at 0.1488. Polish may simply have the least rule complexity to learn, but it may also exhibit a coarse grained annotation scheme, resulting in fewer rule types. German may have an intrinsically higher rule complexity, but it is more likely that the high ratio is related to the flat annotation scheme. The number of unique constituent types ranges from eight on the low end (Korean and Swedish) to upwards of 65 for Hebrew.

Looking at the test sets, we see a noticeable increase in the ratios. This is to be expected due to the much smaller sizes, but for some languages such as Hebrew and Polish, there is no substantial increase in the ratios. Other languages, such as Swedish, Korean, French and especially German, see rather large increases in the ratio; for German, the rule types make up a third of all rules, pointing to a high number of unique rules. For completeness we also provide the F-scores. Note that these cannot be compared to previous work because of the small test set size.

5 Known Rules

We first have a closer look at *known rules* used in the parse trees. By known rules we mean rules that have been seen in the training data, but not necessarily for the sentence under consideration. I.e., we are interested in how often the parser produces known rules and how often those are correct or wrong.

Table 2 shows statistics about known rules. Row 1 gives the absolute number of rules in the gold test sentences, and row 2 the equivalent for the parsed trees. Row 3 gives the percentage of known rules that are predicted correctly, i.e., are also observed in the gold tree of the test sentence. Row 4 shows the percentage of known rules that were incorrect[7]. Row 5 shows the percentage of cases from row 4 where the incorrect rule has the correct yield (i.e. covers the same words). Row 6 shows the corresponding percentage of incorrect rules with incorrect yield.

We observe that the numbers of rules in the parsed trees are very similar to the numbers in the gold trees. The highest difference occurs for Hebrew[8], with 182 more rules in the gold trees, and for German, with 181. However, the ratio of known rules that were predicted correctly (row 3) varies considerably: On the one extreme, Polish shows a ratio of 95.54%, in contrast to German, which shows only 73.03%, and to Korean's 77.01%. Overall, however, all of these ratios are above 70%; all but German and Korean are above 80%. This tells us that in most cases, known rules are parsed correctly. Row 4 shows the opposite, i.e., the percentage of known rules that were not parsed correctly.

Rows 5 and 6 in Table 2 split the incorrectly parsed known rules into cases where the yield of

[6]This information was collected and is available along with code to extract rules for individual treebanks at https://github.com/ddakota/neural_parser_span_productivity.

[7]We note that percentages in rows 3/4 and 5/6 do not add to 1 due to unary nodes.

[8]For better readability, we use the terms language and treebank indiscriminately since we look at one treebank per language.

	Train5k					Test500					Eval
Language	Total	Type	Ratio	Avg Len	Const	Total	Type	Ratio	Avg Len	Const	F-Score
Arabic	226 726	4 214	0.0186	44.68	22	17 368	917	0.0528	35.31	18	87.38
Basque	72 090	3 431	0.0476	12.38	11	6 917	672	0.0972	11.70	11	90.47
Chinese	129 500	2 064	0.0159	24.56	28	8 530	415	0.0487	23.01	19	89.49
English	93 585	4 355	0.0465	24.01	26	9 356	1 038	0.1110	23.78	19	92.68
French	101 464	7 942	0.0782	30.20	29	10 155	1 454	0.1432	30.06	27	85.40
German	36 754	5 468	0.1488	17.57	23	3 555	1 157	0.3255	17.25	20	85.73
Hebrew	257 381	3 788	0.0147	25.61	65	23 034	854	0.0371	23.02	56	93.77
Hungarian	102 827	5 670	0.0551	22.00	15	8 916	906	0.1016	18.49	15	94.07
Korean	73 816	2 844	0.0385	11.56	8	7 977	911	0.1142	12.37	8	82.75
Polish	134 549	564	0.0042	10.18	35	13 675	295	0.0220	10.28	33	95.66
Swedish	62 214	3 694	0.0594	15.27	8	6 240	887	0.1421	15.37	8	88.82

Table 1: Rule statistics for training and test sets. Total is all found rules, Type is the unique number of rules, while Ratio is type/total ratio of rules. Avg Len is the average number of words per sentence and Const is the set of constituent labels.

	Ara.	Bas.	Chi.	Eng.	Fre.	Ger.	Heb.	Hun.	Kor.	Pol.	Swe.
1) No. gold rules	17 368	6 917	8 530	9 356	10 155	3 555	23 034	8 916	7 977	13 675	6 240
2) No. parsed rules	17 347	6 922	8 461	9 340	10 119	3 374	23 216	8 899	7 998	13 669	6 147
3) % parsed $\cap$ train $\cap$ gold	89.47	86.88	90.85	89.40	80.44	73.03	93.29	84.58	77.01	95.54	84.32
4) % parsed $\cap$ train $\cap$ $\neg$ gold	8.77	9.17	6.93	7.51	11.94	12.95	5.40	6.04	20.51	3.82	10.85
5) 4. + correct yield	61.10	64.72	60.75	69.61	56.21	51.72	74.62	63.38	72.07	82.18	56.72
6) 4. + wrong yield	38.90	35.28	39.25	30.39	43.79	48.28	25.38	36.62	27.93	17.82	43.28

Table 2: Known rules in the parsed test sentences.

the rule is correct and where the yield is not correct. The former are cases where the parser found a correct constituent, but either the constituent label is wrong, or the internal structure in terms of daughter nodes is incorrect. The latter are the more concerning cases where the parser did not find a correct constituent. This could indicate that the parser is too conservative in the sense that it has a tendency to overuse known rules instead of creating novel ones. The statistics show that there is a significant amount of parsed rules that follow this pattern. Polish and Hebrew have the fewest with 17.82% and 25.38%. On the other end is German with 48.28%, i.e., almost half of all incorrectly parsed known rules are structurally wrong. Next in line are French and Swedish with percentages of around 43%. Note that these languages are also the languages with the highest ratio of unique rules in the test data (see Table 1) while Polish and Hebrew have the lowest ratios.

Overall, most of the known rules are parsed correctly, and most of the incorrectly parsed rules have the correct yield. However, there are considerable differences across languages. The ratio of incorrect rules with the wrong yield, the more detrimental type of error, seems to be correlated with the ratio of unique rules.

6 Novel Rules

Now we turn to our primary interest: The parser's ability to produce novel rules, i.e., rules that have not been seen in training, and could not be produced by a traditional, grammar-based CYK parser. We distinguish between cases where those novel rules are correct (section 6.1) and where they are incorrect (section 6.2).

6.1 Correct Novel Rules

Here, we focus on novel rules that were parsed correctly. This gives us an indication to what degree the parser can use context information to create the correct rule. I.e., for every sentence, we identify if there are gold rules not found in the training set and then determine whether the parser can create these novels rules. The results are shown in Table 3.

One instant observation is that for most languages, the parser produces (many) more novel rules than in the gold trees, indicating the ability of the parser to develop novel rules. The exceptions are Korean, for which the parser produces fewer novel rules, and Hungarian, for which the parser produces only slightly more novel rules (381 vs. 364). There is, however, a significant variance between languages when we look at how many of the novel rules are correct. Both German and Hungar-

Language	Gold	Parsed	Correct	Most freq. const	Avg. no. daughters
Arabic	242	305	41	VP (31), NP (4), S (3)	VP (3.42), NP (1.79), S (2.08)
Basque	188	273	47	S (34), SP (5), SN (4)	S (3.09), SP (1.55), SN (1.63)
Chinese	47	188	11	VP (5), NP (5), IP (1)	VP (2.18), NP (1.78), IP (2.22)
English	255	301	92	NP (41), VP (33), S (7)	NP (2.21), VP (2.31), S (1.73)
French	552	771	166	SENT (67), NP (36), VN (15)	SENT (5.91), NP (2.40), VN (1.71)
German	442	473	209	S (148), VP (20), PP (19)	S (3.80), VP (2.51), PP (2.70)
Hebrew	206	305	49	S (26), S-ROOT (19), NP (2)	S (3.13), S-ROOT (5.02), NP (1.72)
Hungarian	364	381	204	CP (195), NP (6), XP (2)	CP (4.86), NP (2.04), XP (3.65)
Korean	244	199	27	NP (23), VP (3), AUXP (1)	NP (1.56), VP (2.04), AUXP (1.86)
Polish	14	87	2	fpt (1), fno (1)	fpt (1.19), fno (1.45)
Swedish	265	298	94	S (63), NP (16), VP (12)	S (3.26), NP (1.76), VP (2.44)

Table 3: Correctly parsed novel rules. Gold and Parsed are the number of rules not found in train5k and Most freq. const and avg. no. daughters show the top three constituent labels for rules from the parse trees.

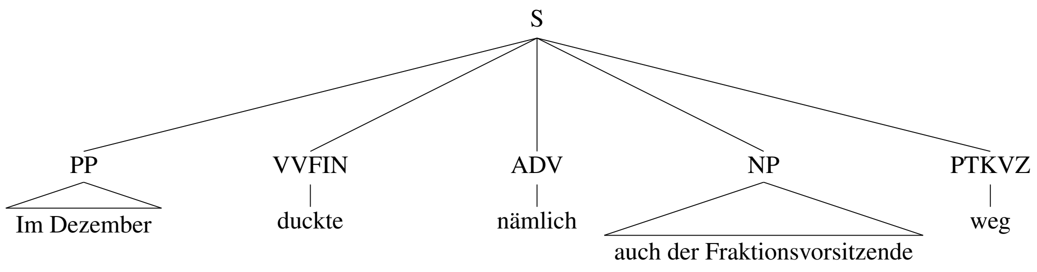

Figure 1: Correct novel S rule from the German trebank. (Eng.: In December even the chairman of the parliamentary group sidestepped.)

ian are on the high end with a match rate of 47.3% (209/442) and 56% (204/364) correct respectively, while most other languages range between 13% and 30%. Polish is an extreme case, it has only 14 novel rules in the gold trees, which makes sense in the light of the statistics in Table 1, showing that Polish has short sentences (on average 10.28 words) and a low ratio of unique rules (0.0220). Additionally, Polish has the highest ratio of correct gold rules and incorrect rules with the correct yield (see Table 2). However, the parser still creates 87 novel rules, only two of which match the gold rules. These results may indicate that the parser's creativity would benefit from added constraints in certain situations.

Returning to Table 3, the average number of daughters in parse trees[9] can be seen as an indication of how flat or hierarchical the parser prefers its trees. When examining the most frequent constituent labels in novel rules from the parse trees, most languages show one particular label as the dominant source of correct novel rules, and for almost all it is either an S type or an NP, the exception being Arabic, where it is a VP. With regard to the daughters, Polish prefers very short phrases. However, for both German and Hungarian, the languages with the highest ratio of correctly parsed novel rules, the most frequent label (S and CP respectively) has many daughters while the other two labels have considerably fewer. This seems to be an indication that S and CP collect unattached constituents in the respective annotation schemes. Figure 1 shows an example of a German sentence collecting constituents under the S node.

6.2 Incorrect Novel Rules

Next we look at novel rules that are parsed incorrectly. First, we identify novel rules in every gold test sentence, and if a novel rule is found, we see if it is also found in the parsed tree for that sentence. If the parser produces a novel rule not found in the corresponding gold test sentence (i.e., it is incorrect), we group it into one of four types of errors. We explicitly exclude sentences in which the parsed tree is incorrect, but the rule is known[10].

[9]Gold trees in the test set are very similar in terms of average number of daughters, the only exceptions are English and Swedish S nodes, which are considerably longer in the gold test trees. German also shows slight variation.

[10]The rule, however, may exist in another test gold sentence, but does not exist in the current gold test sentence.

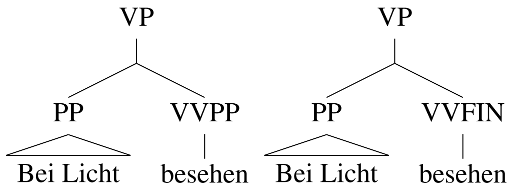

Figure 2: Type 1 error (gold left, parse right; Eng.: In the cold light of day).

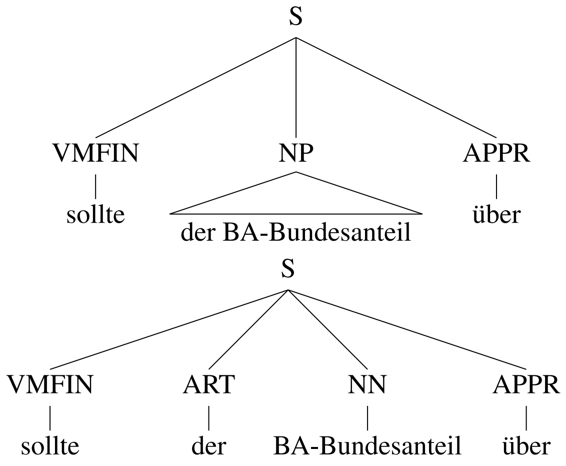

Figure 3: Type 2 error (gold top, parser bottom; Eng.: should the BA federal portion [be] above . . .).

We focus on this type of error since we assume the parser will prefer known rules (see Table 2), and such situations where the parser produces a completely novel span can give us insights into where the parser's creativity needs to be constrained or modified.

Error Types Since the annotation schemes across the languages are very different, defining error types that generalize across treebank annotations is difficult. We define four error types that are relevant on a rule level, but also collect additional statistics that allow a glimpse of the treebank specific errors. The four types of errors are:

T1 **Label errors:** The parser covers the same yield as the gold tree and has the same daughters, but at least one label (either mother or daughter, including POS daughters) is different (see Figure 2 for an example)[11].

T2 **Flatness errors:** The parsed rule covers the same yield but contains more daughters (i.e., it is flatter than the gold tree (Figure 3).

[11]The examples are taken from the German treebank.

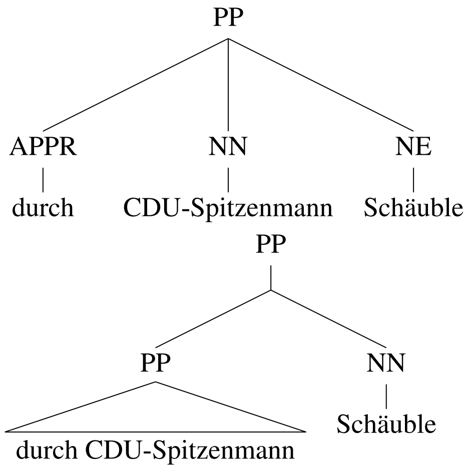

Figure 4: Type 3 error (gold top, parser bottom; Eng.: by CDU head Schäuble).

T3 **Hierarchy errors:** The parsed rule covers the same yield but contains fewer daughters (i.e., it is more hierarchical) than the gold tree (Figure 4).

T4 **Unmatched errors:** The parser covers a yield for which there is no corresponding gold rule (in terms of yield; Figure 5).

Note that the error types are strictly based on single rules (and their yields). We do not make claims about errors in other parts of the trees, including the subtrees below the daughters, though we assume that the parser has selected the rule at this level in the tree based on the composition of the daughters up to this projection layer. We also need to point out that some errors fit into more than one type and are thus counted more than once.

Analysis Table 4 shows an overview of the error types in novel rules across the languages. We first see that almost a quarter of the total errors across the treebanks can be viewed as label errors (T1), further suggesting that actual identification of novel rules is quite successful. On the low end, this type constitutes around 16% of total errors for French (though the overall number of errors is high for French). On the high end, Korean has around 60%, and most languages fall between 20-30% of label errors. Given that many rules in the test set are unique, it is to be expected that a large number of errors involves wrong labels, but the parser seems to be able to identify novel rules. For Korean, the parser favors NP labels (with about 2/3 of the label errors being NPs), potentially because these are the

[VROOT . . . [S [CPP [PP [APPR von] [ADV da] [APZR an]] [KON und] [PP [APPR bis] [CARD 1987]]] [ADV jedoch] [AP [AVP [ADV nur] [ADV noch]] [CARD 46]]] [$. .]]

[VROOT . . . [PP [APPR von] [ADV da] [APZR an]] [KON und] **[AP [PP [APPR bis] [CARD 1987]] [ADV jedoch] [AP [ADV nur] [ADV noch] [CARD 46]]]** [$. .]]

Figure 5: Type 4 error (marked in bold; Eng.: but from then on and till 1987 only 46).

Lang.	Rules	T1	T2	T3	T4	Total
Arabic	154	39	78	17	53	187
Basque	141	36	29	23	56	144
Chinese	60	24	17	7	21	69
English	137	46	49	15	29	139
French	506	87	142	89	225	543
German	231	55	122	39	55	271
Hebrew	149	52	105	15	38	210
Hungarian	140	36	109	15	37	197
Korean	103	64	12	4	25	105
Polish	10	4	4	2	2	12
Swedish	148	31	72	9	36	148
Total	1 786	472	739	241	571	2 033

Table 4: Error types for novel rules per language (T1: label errors, T2: flatness errors, T3: hierarchy errors, T4: unmatched errors).

most frequent type of unique rules.

Across the board, flatness errors (T2) are more common than hierarchy errors (T3), and German, Hebrew, Hungarian, and Swedish are heavily skewed towards flatness errors. This suggests that the parser prefers flatter tree structures. For German, this is understandable since the annotation scheme does not project to the noun phrase level inside a prepositional phrase so that the parser needs to learn from context when it needs an NP node and when it does not.

Another interesting observation concerns the difference between the number of rules and the total number of errors. If the difference is high, then many rule errors fall into more than one error type. Basque, English, Korean, Polish, and Swedish have very few multiple errors. However, for Hebrew and Hungarian, there are almost 50% errors falling into more than one type. These languages are both polarized towards flatness errors, suggesting that the parser's preference for flatter rules results in multiple error types; a trend that Arabic, French, and German also demonstrate. Swedish has no double errors. This strongly suggests that the wrong projections in Swedish are higher up in the tree and do not percolate upwards. This is supported by the fact that S is the type of constituent with the highest count of incorrect novel rules (see Table 5).

We also see the prevalence of unmatched errors (T4) across languages, i.e., errors that do not have any correspondence in the gold tree. French is an extreme outlier on the upper end with around 40% of errors of this type. This seems to be due to a single constituent type, multi-word expressions (MWEs; see Table 5). The parser seems to overgeneralize this constituent to cover sequences of numbers.

Table 5 shows the most frequent error types per constituent type[12]. Most languages fall into one of two categories: those with many S type phrase errors (Basque, French, German, Hebrew, Hungarian, and Swedish) and those with more balanced error types, with potentially one or two outliers (Chinese, English, and Korean). This is a continuation of the results in Table 3 in that these languages also possess high unique ratios of S phrases, thus are also prone to errors on the same phrases.

In contrast, Arabic shows a distinct trend of having primarily VP errors, as it did with correct VP rules. When looking at individual treebank statistics, Arabic has, on average, the longest VP rules in terms of number of daughters, at over 3.23 in the training sentences and 3.42 in the test sentences. (All other language are below 3, with an average around 2.) The longest Arabic VP has 28 daughters. As a consequence of the length of Arabic VPs, the parser seems to prefer flatter parses that attach more constituents directly to a VP. While just under 10% of the VPs in the training set are unique, this increases to around 25% in the test set. However, this factor alone cannot explain the VP errors since German shows a unique VP ratio of about 20% in the training set and 44% in the test set, but does not have such a large number of VP errors.

One possible explanation can be found in the fact that Arabic has free word order, which means that many different configurations are possible under the VP node.

[12]We exclude Polish because of its small number of errors.

	Arabic	Basque	Chinese	English	French	German	Hebrew	Hungarian	Korean	Swedish
label	VP (11)	S (16)	NP (9)	NP (26)	SENT (20)	S (26)	S (10)	CP (28)	NP (41)	S (13)
	ADJP (6)	GRUP.NOM (6)	VP (5)	VP (7)	NP (19)	NP (8)	NP (9)	NP (4)	ADJP (11)	NP (6)
	NP (5)	GV (5)	ADVP (3)	PP (4)	PP (7)	CNP (5)	S-ROOT (7)	ADJP (2)	VP (8)	PP (4)
flatness	VP (37)	S (27)	VP (6)	VP (18)	SENT (55)	S (84)	S-ROOT (65)	CP (100)	VP (6)	S (38)
	S (17)	SN (1)	6 (NP)	NP (16)	NP (25)	PP (14)	S (15)	NP (6)	NP (4)	NP (12)
	NP (11)	SADV (1)	CP (3)	S (9)	PP (14)	VP (11)	NP (12)	XP (3)	ADJP (2)	VP (8)
hierarchy	VP (12)	S (15)	IP (4)	VP (6)	SENT (34)	S (22)	S-ROOT (10)	CP (15)	NP (4)	S (6)
	S (4)	SP (3)	NP (2)	NP (3)	NP (22)	PP (5)	S (4)			XP (1)
	NP (1)	SADV (2)	VP (1)	S (3)	Sint (11)	VP (5)	VP (1)			NP (1)
unmatched	NP (17)	SN (13)	VP (6)	VP (8)	DET+ (60)	PP (19)	S (9)	CP (21)	NP (14)	S (9)
	VP (16)	S (12)	NP (4)	NP (6)	NP (46)	NP (9)	NP (5)	NP (9)	ADJP (4)	NP (9)
	PP (7)	GV (10)	ADVP (3)	S (5)	PP (28)	S (8)	PP (4)	ADJP (3)	AUXP (3)	VP (7)

Table 5: The most frequent error types per constituent type.

7 Conclusion & Future Work

We have started the first (to our knowledge) multilingual error analysis of how a span-based neural parser behaves across languages and treebanks, with a focus on the parser's creativity. We were able to identify some generalized behaviors but also its interaction with treebank-specific annotations.

Overall, the parser is good at creating new rules, and incorrect parses for novel rules tend towards the less serious errors (T1). For a range of languages, however, the parser has a tendency to overgenerate novel rules, including languages that have a large ratio of unique rules. It also has a tendency to create flat structures, which tend to collect constituents it could not group otherwise.

Consequently, the parser may benefit from more constraints to reduce over-generations and preferences for flatter structures, but it may be difficult to generalize constraints across multiple treebank annotations. Areas worth exploring are examining how the rule errors change over the development set within each epoch of training to better understand where the parser is quickly able to optimize for specific rule and label types, and where the parser shifts its preferences in order to increase performance, and the subsequent trade-offs. Another approach may be using grammatical functions as natural constraints that provide additional information to distinguish between structures.

Additionally, it would be worthwhile to examine treebanks for the same language that contain different annotation schemes to determine whether the effects we found are due to language or treebank characteristics.

Another more theoretical angle is examining what exaclty is encoded in the pretrained language models used in the parsers. Probing techniques have been applied in dependency parsing (Hewitt and Manning, 2019; Kulmizev et al., 2020) to extract trees from language models such as BERT and ELMO (Peters et al., 2018). Such avenues of research could be extended to constituency parsing.

Acknowledgements

The authors would like to thank the members of the Uppsala NLP Parsing Group: Joakim Nivre, Sara Stymne, Artur Kulmizev and Ali Basirat, as well as the reviewers for their comments. The first author is supported by the Swedish strategic research programme eSSENCE.

References

Sabine Brants, Stefanie Dipper, Peter Eisenberg, Silvia Hansen, Esther König, Wolfgang Lezius, Christian Rohrer, George Smith, and Hans. Uszkoreit. 2004. TIGER: Linguistic Interpretation of a German Corpus. *Journal of Language and Computation*, 2004 (2):597–620.

Michael Collins. 1999. *Head-Driven Statistical Methods for Natural Language Parsing*. Ph.D. thesis, University of Pennsylvania.

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.

Daniel Dakota. 2018. *Using Distributional Word Representations for the Statistical Constituency Parsing of German*. Ph.D. thesis, Indiana University.

Daniel Dakota and Sandra Kübler. 2017. Towards Replicability in Parsing. In *Proceedings of the International Conference Recent Advances in Natural Language Processing, RANLP 2017*, pages 185–194, Varna, Bulgaria.

David Gaddy, Mitchell Stern, and Dan Klein. 2018. What's going on in neural constituency parsers? an analysis. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 999–1010, New Orleans, Louisiana.

David Hall, Greg Durrett, and Dan Klein. 2014. Less grammar, more features. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics)*, pages 228–237, Baltimore, Maryland.

John Hewitt and Christopher D. Manning. 2019. A structural probe for finding syntax in word representations. 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 4129–4138, Minneapolis, Minnesota.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 8(9):1735–1780.

Mija Kim and Jungyeul Park. 2020. A note on constituent parsing for Korean. *Natural Language Engineering*, page 1–24.

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.

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*, pages 2676–2686, Melbourne, Australia.

Dan Klein and Christopher Manning. 2003. Accurate Unlexicalized Parsing. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics*, pages 423–430, Sapporo, Japan.

Daniël de Kok, Jianqiang Ma, Corina Dima, and Erhard Hinrichs. 2017. PP attachment: Where do we stand? In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics*, pages 311–317, Valencia, Spain.

Sandra Kübler. 2005. How Do Treebank Annotation Schemes Influence Parsing Results? Or How Not to Compare Apples And Oranges. In *Proceedings of the International Conference on Recent Advances in Natural Language Processing RANLP 2005*, pages 293–300, Borovets, Bulgaria.

Sandra Kübler, Ryan McDonald, and Joakim Nivre. 2009. *Dependency Parsing*. Morgan Claypool.

Artur Kulmizev, Vinit Ravishankar, Mostafa Abdou, and Joakim Nivre. 2020. Do neural language models show preferences for syntactic formalisms? In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4077–4091, Online.

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.

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*, pages 98–103, Sofia, Bulgaria.

Roger Levy and Christopher Manning. 2003. Is It Harder to Parse Chinese, or the Chinese Treebank? In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics*, pages 439–446, Sapporo, Japan.

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

Quy Nguyen, Yusuke Miyao, Hiroshi Noji, and Nhung Nguyen. 2018. An empirical investigation of error types in Vietnamese parsing. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 3075–3089, Santa Fe, New Mexico.

Joakim Nivre. 2003. An efficient algorithm for projective dependency parsing. In *Proceedings of the Eighth International Conference on Parsing Technologies*, pages 149–160, Nancy, France.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. 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 2227–2237, New Orleans, Louisiana.

Slav Petrov and Dan Klein. 2007. Improved Inference for Unlexicalized Parsing. In *Proceedings of Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics*, pages 404–411, Rochester, NY.

Ines Rehbein and Josef van Genabith. 2007. Treebank Annotation Schemes and Parser Evaluation for German. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 630–639, Prague, Czech Republic.

Kenji Sagae and Alon Lavie. 2005. A classifier-based parser with linear run-time complexity. In *Proceedings of the Ninth International Workshop on Parsing Technology*, pages 125–132, Vancouver, British Columbia.

Djamé Seddah, Sandra Kübler, and Reut Tsarfaty. 2014. Introducing the SPMRL 2014 Shared Task on Parsing Morphologically-rich Languages. In *Proceedings of the First Joint Workshop on Statistical Parsing of Morphologically Rich Languages and Syntactic Analysis of Non-Canonical Languages*, pages 103–109, Dublin, Ireland. Dublin City University.

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.

Satoshi Sekine and Michael Collins. 1997. EVALB Bracket Scoring Program. http://nlp.cs.nyu.edu/evalb/.

Mitchell Stern, Jacob Andreas, and Dan Klein. 2017. A Minimal Span-Based Neural Constituency Parser. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, pages 818–827, Vancouver, Canada.

Naiwen Xue, Fei Xia, Fu-Dong Chiou, and Palmer Marta. 2005. The Penn Chinese TreeBank: Phrase structure annotation of a large corpus. *Natural Language Engineering*, 11(2):207–238.