Squib

A Novel Alignment-based Approach for PARSEVAL Measures

Eunkyul Leah Jo[†]
The University of British Columbia
Department of Computer Science
eunkyul@student.ubc.ca

Angela Yoonseo Park[†]
The University of British Columbia
Department of Linguistics
apark03@student.ubc.ca

Jungyeul Park[*]
The University of British Columbia
Department of Linguistics
jungyeul@mail.ubc.ca

We propose a novel method for calculating PARSEVAL measures to evaluate constituent parsing results. Previous constituent parsing evaluation techniques were constrained by the requirement for consistent sentence boundaries and tokenization results, proving to be stringent and inconvenient. Our new approach handles constituent parsing results obtained from raw text, even when sentence boundaries and tokenization differ from the preprocessed gold sentence. Implementing this measure is our evaluation by alignment approach. The algorithm enables the alignment of tokens and sentences in the gold and system parse trees. Our proposed algorithm draws on the analogy of sentence and word alignment commonly used in machine translation (MT). To demonstrate the intricacy of calculations and clarify any integration of configurations, we explain the implementations in detailed pseudo-code and provide empirical proof for how sentence and word alignment can improve evaluation reliability.

[†]Equal contribution.

[*]Corresponding author.

Action Editor: Michael White. Submission received: 4 May 2023; revised version received: 10 November 2023; accepted for publication: 13 December 2023.

https://doi.org/10.1162/coli_a_00512

1. Introduction

Evaluation is a systematic method for assessing a design or implementation to measure how well it achieves its goals. In natural language processing (NLP) systems, quality is assessed using evaluation criteria and measures by comparing them to gold standard answer keys. In the context of constituent parsers, we evaluate the fitness of our predicted parse tree against the human-labeled reference parse tree in the test set. For constituent parsing, whether statistical or neural, we rely on the EVALB implementation.[1] It uses the PARSEVAL measures (Black et al. 1991) as the standard method for evaluating parser performance. A constituent in a hypothesis parse of a sentence is labeled as correct if it matches a constituent in the reference parse with the same nonterminal symbol and span (starting and end indexes). Despite its success in evaluating language technology, EVALB faces an unresolved critical issue in our discipline. EVALB has constraints, such as requiring the same tokenization results. Its implementation assumes equal-length gold and system files, with one tree per line. Nevertheless, we evaluate parser accuracy using EVALB's standard F1 metric for constituent parsing.

Furthermore, in today's component-based NLP systems, it is common practice to evaluate parsers individually. This approach helps improve accuracy by preventing errors from propagating through dependent preprocessing steps. We propose a novel method for measuring PARSEVAL in constituent parsing evaluation, which more accurately simulates real-world scenarios and extends beyond controlled and task-specific settings. Hence, we propose a new way of calculating PARSEVAL measures, which aims to solve some limitations of EVALB for more error-free and accurate evaluation metrics. By rectifying its restrictions, we would be able to present refined precision and recall for the F1 measures in constituent parsing evaluation.

To emphasize the importance of our new methodology, we will first address the task-specific inherent problems in tokenization and sentence boundary detection before constituent parsing. We will then demonstrate the new implementation of PARSEVAL measures by presenting solutions to each identified mismatch case and their corresponding algorithms. To ensure the reliability and applicability of these algorithms, we will also conduct additional discussion towards the end of the squib.

2. Known Problems

To illustrate how we present this new approach, consider some known problems of EVALB that dictate why this new solution is needed. Firstly, evaluation cannot be complete if the terminal nodes of the gold and system trees are different, causing a word mismatch error. An example of this can be found when the gold and system spans differ on the character level with tokens like *This* versus *this*. These tokens are considered identical if we disregard the distinction made by letter case. Hence, we can resolve this character discrepancy by converting all letters to lowercase. This adjustment allows our evaluation system to treat *This* and *this* as a matching word pair.

Secondly, tokens represented as terminal nodes in gold parse trees can differ from those in parser outputs due to the token and sentence segmentation of the system. During preprocessing, even with the same sentence boundary, tokenization discrepancies may arise when compared to the gold standard tree from the Penn Treebank.

1 http://nlp.cs.nyu.edu/evalb. There is also an EVALB_SPMRL implementation, specifically designed for the SPMRL shared task (Seddah et al. 2013; Seddah, Kübler, and Tsarfaty 2014).

This mainly occurs when periods and contractions create ambiguity among words that are abbreviations or acronyms. Such discrepancies can lead to the preprocessing results diverging into several different tokenization schemes. Importantly, EVALB is unable to evaluate constituent parsing when the system's tokenization result differs from the gold standard.

Example: gold *This ca␣n't be right␣.*
system *this can␣not be right␣.* where ␣ is a token delimiter.

The discrepancy is evident in such a comparison of *ca␣n't* (gold) versus *can␣not* (system) for *cannot*. In this context, it is readily apparent to human eyes that the gold and system tokens are actually the same. To handle such an instance that EVALB cannot manage, we observe that *ca␣n't* and *can␣not* are indifferent to each other between all tokens when we create the set of constituents. This observation plays a pivotal role in shaping our approach to address tokenization challenges, and it is equally significant in resolving issues related to sentence boundaries prior to constituent parsing.

The mission of a sentence boundary detection system is to recognize where each sentence starts and ends. A major hurdle in this task is to detect sentence beginnings and endings given some text that lacks punctuation marks. In the following example, although there is no tokenization discrepancy, a sentence boundary discrepancy exists. In the system, *Click here To view it.* is perceived as two separate sentences: *Click here* and *To view it*. The previous method proposed by EVALB could not assign a score to the unmatched sentences. However, it is worth noting that there are partial matches between the gold and system trees, even though the current EVALB does not consider them.

Example: gold *Click here To view it␣.*
system *Click here ⊓ To view it␣.* where ⊓ is a sentence delimiter.

Consequently, tokens undergoing tokenization and sentences handled through sentence boundary detection share a common quality during evaluation. The gold and system results turn out to be two identical sequences of characters. However, they may still differ in length across tokens and lines due to the various tokenization and sentence boundary detection results. Therefore, we suggest the next step beyond EVALB, re-indexing system lines through sentence and word alignment. As part of our solution, we propose an evaluation-by-alignment algorithm to avoid mismatches in sentences and words when deriving constituents for eventual evaluation. The algorithms of the new PARSEVAL measures allow us to reassess such edge cases of mismatch.

Finally, the question of how to evaluate constituent parsing results from these end-to-end systems has been a longstanding challenge. Conventionally, EVALB has proven useful in a component-based preprocessing pipeline, with each component evaluated individually under ideal circumstances. However, conducting end-to-end evaluations with all preprocessing in a single pipeline can offer an alternative perspective in constituent parsing evaluation, and this is the approach adopted for the proposed new PARSEVAL measures. By addressing the constraints discovered in EVALB that lead to issues in preprocessing, we create an opportunity to compare end-to-end parser results. Even when different preprocessing results are produced due to the use of various models in sentence boundary detection and tokenization, the extension of the evaluation

technique with the new way of calculating PARSEVAL measures makes this comparison possible.

3. Implementing New PARSEVAL Measures

3.1 Algorithm

To describe the proposed algorithms, we use the following notations for conciseness and simplicity. $\mathcal{T}_{\mathcal{L}}$ and $\mathcal{T}_{\mathcal{R}}$ introduce the entire parse trees of gold and system files, respectively. $\mathcal{T}_{\mathcal{L}}$ is a simplified notation representing $\mathcal{T}_{\mathcal{L}(l)}$, where l is the list of tokens in $\mathcal{L}$. This notation applies in the same manner to $\mathcal{R}$. $\mathcal{S}_{\mathcal{T}}$ represents a set of constituents of a tree $\mathcal{T}$, and $\mathcal{C}(\mathcal{T})$ is the total number of constituents of $\mathcal{T}$. $\mathcal{C}(\texttt{tp})$ is the number of true positive constituents where $\mathcal{S}_{\mathcal{T}_{\mathcal{L}}} \cap \mathcal{S}_{\mathcal{T}_{\mathcal{R}}}$, and we count it per aligned sentence. The presented Algorithm 1 demonstrates the pseudo-code for the new PARSEVAL measures.

In the first stage, we extract leaves $\mathcal{L}$ and $\mathcal{R}$ from the parse trees and align sentences to obtain $\mathcal{L}'$ and $\mathcal{R}'$ using Algorithm 2. While the necessity of sentence alignment is rooted in a common phenomenon in cross-language tasks such as machine translation, the intralingual alignment between gold and system sentences does not share the same necessity because $\mathcal{L}$ and $\mathcal{R}$ are identical sentences that only differ in sentence boundaries and tokenization results. A notation $\not\sqcup$ is introduced to represent spaces that are removed during sentence alignment when comparing $\mathcal{L}_i$ and $\mathcal{R}_j$, irrespective of their tokenization results. If there is a mismatch due to differences in sentence boundaries, the algorithm accumulates the sentences until the next pair of sentences represented as CASE n $(i+1, j+1)$, is matched. In the next stage of Algorithm 1, we align trees based on $\mathcal{L}'$ and $\mathcal{R}'$ to obtain $\mathcal{T}_{\mathcal{L}'}$ and $\mathcal{T}_{\mathcal{R}'}$. By iterating through $\mathcal{T}_{\mathcal{L}'}$ and $\mathcal{T}_{\mathcal{R}'}$, we conduct word alignment and compare pairs of sets of constituents for each corresponding pair

Algorithm 1 Pseudo-code for new PARSEVAL measures

function PARSEVALMEASURES ($\mathcal{T}_{\mathcal{L}}$ and $\mathcal{T}_{\mathcal{R}}$):
  Extract the list of tokens $\mathcal{L}$ and $\mathcal{R}$ from $\mathcal{T}_{\mathcal{L}}$ and $\mathcal{T}_{\mathcal{R}}$
  $\mathcal{L}', \mathcal{R}' \leftarrow$ SENTENCEALIGNMENT$(\mathcal{L}, \mathcal{R})$ where LEN$(\mathcal{L}')$ = LEN$(\mathcal{R}')$
  Align trees based on $\mathcal{L}'$ and $\mathcal{R}'$ to obtain $\mathcal{T}_{\mathcal{L}'}$ and $\mathcal{T}_{\mathcal{R}'}$
  while $\mathcal{T}_{\mathcal{L}'}$ and $\mathcal{T}_{\mathcal{R}'}$ **do**
    Extract the list of tokens l and r from $\mathcal{T}_{\mathcal{L}'_i}$ and $\mathcal{T}_{\mathcal{R}'_i}$
    $l', r' \leftarrow$ WORDALIGNMENT(l, r)
    $\mathcal{S}_{\mathcal{T}_{\mathcal{L}}} \leftarrow$ GETCONSTITUENT$(\mathcal{T}_{\mathcal{L}'_i(l')}, 0)$ where $0 < i \leq$ LEN$(\mathcal{L}')$
    $\mathcal{S}_{\mathcal{T}_{\mathcal{R}}} \leftarrow$ GETCONSTITUENT$(\mathcal{T}_{\mathcal{R}'_i(r')}, 0)$ where $0 < i \leq$ LEN$(\mathcal{R}')$
    $\mathcal{C}(\mathcal{T}_{\mathcal{L}}) \leftarrow \mathcal{C}(\mathcal{T}_{\mathcal{L}})+$ LEN$(\mathcal{S}_{\mathcal{T}_{\mathcal{L}}})$
    $\mathcal{C}(\mathcal{T}_{\mathcal{R}}) \leftarrow \mathcal{C}(\mathcal{T}_{\mathcal{R}})+$ LEN$(\mathcal{S}_{\mathcal{T}_{\mathcal{R}}})$
    while $\mathcal{S}_{\mathcal{T}_{\mathcal{L}}}$ and $\mathcal{S}_{\mathcal{T}_{\mathcal{R}}}$ **do**
      If (LABEL, START$_{\mathcal{L}}$, END$_{\mathcal{L}}$, l'_j) = (LABEL, START$_{\mathcal{R}}$, END$_{\mathcal{R}}$, r'_j) **then**
        $\mathcal{C}(\texttt{tp}) \leftarrow \mathcal{C}(\texttt{tp}) + 1$
      end if
    end while
  end while
  return $\mathcal{C}(\mathcal{T}_{\mathcal{L}})$, $\mathcal{C}(\mathcal{T}_{\mathcal{R}})$, and $\mathcal{C}(\texttt{tp})$

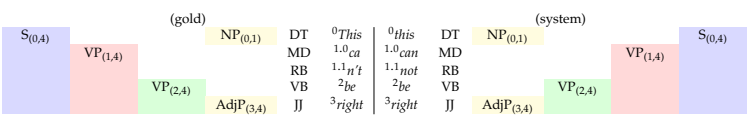
Algorithm 2 Pseudo-code for sentence alignment

function SENTENCEALIGNMENT ($\mathcal{L}$, $\mathcal{R}$):
  while $\mathcal{L}$ and $\mathcal{R}$ **do**
    if $(\mathcal{L}_{i(\not\perp)} = \mathcal{R}_{j(\not\perp)})$ ▷ CASE 1 $_{(i,j)}$
      $\vee\ (\mathcal{L}_{i(\not\perp)} \simeq \mathcal{R}_{j(\not\perp)} \wedge (\mathcal{L}_{i+1(\not\perp)} = \mathcal{R}_{j+1(\not\perp)} \vee \mathcal{L}_{i+1(\not\perp)} \simeq \mathcal{R}_{j+1(\not\perp)}))$ ▷ CASE 2 $_{(i,j)}$
    then
      $\mathcal{L}', \mathcal{R}' \leftarrow \mathcal{L}' + \mathcal{L}_i, \mathcal{R}' + \mathcal{R}_j$ where $0 < i \leq \text{LEN}(\mathcal{L}), 0 < j \leq \text{LEN}(\mathcal{R})$
    else
      while $\neg(\text{CASE 1}_{(i+1,j+1)} \vee \text{CASE 2}_{(i+1,j+1)})$ **do**
        if $\text{LEN}(\mathcal{L}_i) < \text{LEN}(\mathcal{R}_j)$ **then**
          $L' \leftarrow L' + \mathcal{L}_i$
          $i \leftarrow i + 1$
        else
          $R' \leftarrow R' + \mathcal{R}_j$
          $j \leftarrow j + 1$
        end if
      end while
      $\mathcal{L}', \mathcal{R}' \leftarrow \mathcal{L}' + L', \mathcal{R}' + R'$
    end if
  end while
  return $\mathcal{L}', \mathcal{R}'$

of $\mathcal{T}_{\mathcal{L}'_i}$ and $\mathcal{T}_{\mathcal{R}'_j}$. The word alignment in Algorithm 3 follows a logic similar to sentence alignment, wherein words are accumulated in *ll* and *rr* if the pairs of l_i and r_j do not match due to tokenization mismatches. Finally, we extract a set of constituents using Algorithm 4, a straightforward procedure for obtaining constituents from a given tree, which includes the label name, start index, end index, and a list of tokens. The current proposed method utilizes simple pattern matching for sentence and word alignment, operating under the assumption that the gold and system sentences are the same, with minimal potential for morphological mismatches. This differs from sentence and word alignment in machine translation. MT usually relies on recursive editing and EM algorithms due to the inherent difference between source and target languages.

3.2 Examples of Word and Sentence Mismatches

Word mismatch. We have observed that the expression of contractions varies significantly, resulting in inherent challenges related to word mismatches. As the number of contractions and symbols to be converted in a language is finite, we composed an exception list for our system to capture such cases for each language to facilitate the word alignment process between gold and system sentences. In the following example, we achieve perfect precision and recall of 5/5 for both because their constituent trees are exactly matched, regardless of any mismatched words.



Algorithm 3 Pseudo-code for word alignment

```
1: function WordAlignment (l, r):
2:   while l and r do
3:     if (l_i = r_j)                                   ▷ case 1 (i,j)
         ∨ ((l_i ≠ r_j) ∧ (l_{i+1} = r_{j+1}))          ▷ case 2 (i,j)
       then
4:       l', r' ← l_i, r_j where 0 < i ≤ len(l), 0 < j ≤ len(r)
5:     else
6:       while¬(case 1 (i+1,j+1) ∨ case 2 (i+1,j+1)) do
7:         if (len(l) − len(l_0, l_i)) > (len(r) − len(r_0, r_j)) then
8:           ll ← ll + l_i
9:           i ← i + 1
10:        else
11:          rr ← rr + r_j
12:          j ← j + 1
13:        else if
14:      end while
15:      l', r' ← ll, rr
16:    end if
17:  end while
18:  return l', r'
```

Algorithm 4 Pseudo-code for getting constituents

```
1: function GetConstituent (T, start):
2:   if height(T)>2 then
3:     end ← start + len(leaves(T))
4:     S_T ← S_T + (label(T), start, end, leaves(T))
5:     while T do
6:       GetConstituent(T_i, start) where T_i is a child of T
7:       start ← len(leaves(T_i))
8:     end while
9:   end if
10:  return S_T
```

If the word mismatch example is not in the exception list, we perform the word alignment. We can still achieve perfect precision and recall (5/5 for both) without the word mismatch exception list because their constituent trees can be exactly matched based on the word-alignment of $\{^{1.0}ca\ ^{1.1}n't\}$ and $\{^{1.0}can\ ^{1.1}not\}$.

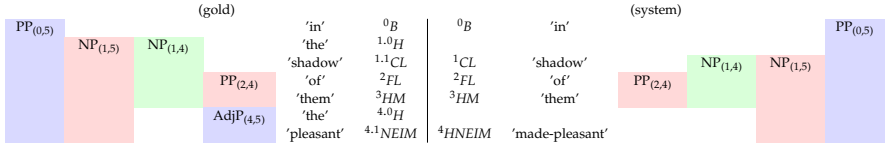
gold	$^{0}This$	$^{1.0}ca$	$^{1.1}n't$	^{2}be	$^{3}right$
system	$^{0}this$	$^{1.0}can$	$^{1.1}not$	^{2}be	$^{3}right$

The effectiveness of the word alignment approach remains intact even for morphological mismatches where "morphological segmentation is not the inverse of concatenation" (Tsarfaty, Nivre, and Andersson 2012), such as in morphologically rich languages.

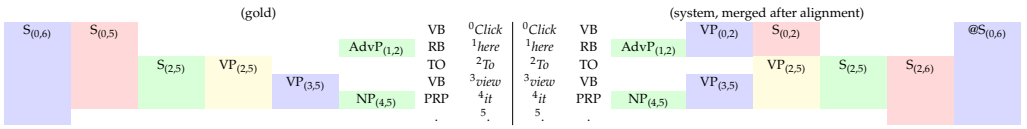
For example, we trace back to the sentence in Hebrew described in Tsarfaty, Nivre, and Andersson (2012) as a word mismatch example caused by morphological analyses:

gold	^{0}B	$^{1.0}H$	$^{1.1}CL$	^{2}FL	^{3}HM	$^{4.0}H$	$^{4.1}NEIM$
	'in'	'the'	'shadow'	'of'	'them'	'the'	'pleasant'
system	^{0}B	^{1}CL		^{2}FL	^{3}HM	$^{4}HNEIM$	
	'in'	'shadow'		'of'	'them'	'made-pleasant'	

Pairs of $\{^{1.0}H\ ^{1.1}CL, ^{1}CL\}$ ('the shadow') and $\{^{4.0}H\ ^{4.1}NEIM, ^{4}HNEIM\}$ ('the pleasant') are word-aligned using the proposed algorithm, resulting in a precision of 4/4 and recall of 4/6.



Sentence mismatch. When there are sentence mismatches, they would be aligned and merged as a single tree using a dummy root node: for example, @S which can be ignored during evaluation. In the following example, we obtain precision of 5/8 and recall of 5/7.



Assumptions. To address morphological analysis discrepancies in the parse tree during evaluation, we establish the following two assumptions: (i) The entire tree constituent can be considered a true positive, even if the morphological segmentation or analysis differs from the gold analysis, as long as the two sentences (gold and system) are aligned and their root labels are the same. (ii) The subtree constituent can be considered a true positive if lexical items align in word alignment, and their phrase labels are the same.

4. Discussion

Complexity. The proposed algorithm has a linear time complexity. Sentence and word alignments require $O(I + J)$, where I and J represent the lengths of the gold and system sentences or words. The process for constituent tree matches uses tree traversal algorithm which requires $O(N + E)$ where N is a number of nodes and E is for branches. We retain the same time complexity of the original `EVALB` by adding alignment-based preprocessing for mismatches of sentences and words.

Comparison. Table 1 compares previous parsing evaluation metrics with the proposed algorithm. `tedeval` (Tsarfaty, Nivre, and Andersson 2012) is based on the tree edit distance of Bille (2005), and numbers of nonterminal nodes in system and gold trees. A similar idea on the tree edit distance was proposed for classifying constituent parsing errors based on subtree movement, node creation, and node deletion (Kummerfeld et al.

Table 1
Comparison to previous parsing evaluation metrics.

	evaluation approach	addressing mismatches
`tedeval`	tree-edit distance based on constituent trees	words
`conllu_eval`	dependency scoring	words and sentences
`sparseval`	dependency scoring	words and sentences
`aligning trees`	constituent tree matches	words
`EVALB`	constituent tree matches	not applicable
proposed method	constituent tree matches	words and sentences

2012). `conllu_eval` for dependency parsing evaluation within Universal Dependencies (Nivre et al. 2020) views tokens and sentences as spans. If there is a mismatch of positions of spans between the system and the gold file on a character level, whichever file has a smaller start value will skip to the next token until there is no start value mismatch. Evaluating sentence boundaries also follows similar processes as tokens. The start and end values of the sentence span are compared between the system and the gold file. When they match, it increases the count of correctly matched sentences. `sparseval` (Roark et al. 2006) uses a head percolation table (Collins 1999) to identify head-child relations between two terminal nodes from constituent parsing trees, and calculate the dependency score. We also add an `aligning trees` method (Calder 1997) in our comparison, which performs an alignment of the tree structures from two different treebanks for the same sentence, both of which utilize distinct POS labels.

A note on constituent parsing. Syntactic analysis in the current field of language technology has been predominantly reliant on dependencies. Semantic parsing in its higher-level analyses often relies heavily on dependency structures as well. Dependency parsing and its evaluation method have their own advantages, such as a more direct representation of grammatical relations and often simpler parsing algorithms. However, constituent parsing maintains the hierarchical structure of a sentence, which can still be valuable for understanding the syntactic relationships between words and phrases. Numerous studies in formal syntax have focused on constituent structures, including combinatory categorial grammar (CCG) parsing (Lewis, Lee, and Zettlemoyer 2016; Lee, Lewis, and Zettlemoyer 2016; Stanojević and Steedman 2020; Yamaki, Taniguchi, and Mochihashi 2023) or tree-adjoining grammar (TAG) parsing (Kasai et al. 2017, 2018). Notably, CCG and TAG inherently incorporate dependency structures. In addition to these approaches, new methods for constituent parsing, such as the linearization parsing method (Vinyals et al. 2015; Fernández-González and Gómez-Rodríguez 2020; Wei, Wu, and Lan 2020), have been actively explored. If a method designed to achieve the goal of creating an end-to-end system utilizes constituent structures, it necessitates more robust evaluation methods for assessing its constituent structure.

5. Conclusion

Despite the widespread use and acceptance of the previous implementation of PARSEVAL measures as the standard tool for constituent parsing evaluation, it has a significant limitation in that it requires specific task-oriented environments. Consequently, there is still room for a more robust and reliable evaluation approach. Various metrics have

attempted to address issues related to word and sentence mismatches by implementing complex tree operations or adopting dependency scoring methods. In contrast, our proposed method aligns sentences and words as a preprocessing step without altering the original PARSEVAL measures. This approach allows us to preserve the complexity of the previous implementation of PARSEVAL while introducing a linear time alignment process. Given the high compatibility of our method with existing PARSEVAL measures, it also ensures the consistency and seamless integration of previous work evaluated using PARSEVAL into our approach. Ultimately, this new measurement approach offers the opportunity to evaluate constituent parsing within an end-to-end pipeline. It addresses discrepancies that may arise during earlier steps, such as sentence boundary detection and tokenization, thus enabling a more comprehensive evaluation of constituent parsing.[2]

Acknowledgments

We are grateful to the action editor Michael White and three anonymous reviewers for their detailed and constructive comments. This research is based on work partially supported by *Students as Partners* for Eunkyul Leah Jo and *The Work Learn Program* for Angela Yoonseo Park at The University of British Columbia.

References

Bille, Philip. 2005. A survey on tree edit distance and related problems. *Theoretical Computer Science*, 337(1):217–239. https://doi.org/10.1016/j.tcs.2004.12.030

Black, Ezra, Steve Abney, Dan Flickinger, et al. 1991. A procedure for quantitatively comparing the syntactic coverage of English grammars. In *Speech and Natural Language: Proceedings*, pages 306–311.

Calder, Jo. 1997. On aligning trees. In *Second Conference on Empirical Methods in Natural Language Processing*, pages 75–80.

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

Fernández-González, Daniel and Carlos Gómez-Rodríguez. 2020. Enriched in-order linearization for faster sequence-to-sequence constituent parsing. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4092–4099. https://doi.org/10.18653/v1/2020.acl-main.376

Kasai, Jungo, Bob Frank, Tom McCoy, Owen Rambow, and Alexis Nasr. 2017. TAG parsing with neural networks and vector representations of supertags. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1712–1722. https://doi.org/10.18653/v1/D17-1180

Kasai, Jungo, Robert Frank, Pauli Xu, William Merrill, and Owen Rambow. 2018. End-to-end graph-based TAG parsing with neural networks. 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 1181–1194. https://doi.org/10.18653/v1/N18-1107

Kummerfeld, Jonathan K., 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.

Lee, Kenton, Mike Lewis, and Luke Zettlemoyer. 2016. Global neural CCG parsing with optimality guarantees. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2366–2376. https://doi.org/10.18653/v1/D16-1262

Lewis, Mike, Kenton Lee, and Luke Zettlemoyer. 2016. LSTM CCG parsing. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 221–231. https://doi.org/10.18653/v1/N16-1026

Nivre, Joakim, Marie-Catherine de Marneffe, Filip Ginter, et al. 2020. Universal Dependencies v2: An evergrowing multilingual treebank collection. In *Proceedings of the 12th Language Resources and Evaluation Conference*, pages 4034–4043.

2 https://github.com/jungyeul/alignment-based-PARSEVAL/.

Roark, Brian, Mary Harper, Eugene Charniak, et al. 2006. SParseval: Evaluation metrics for parsing speech. In *Proceedings of the Fifth International Conference on Language Resources and Evaluation (LREC'06)*, pages 333–338.

Seddah, Djamé, 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.

Seddah, Djamé, Reut Tsarfaty, Sandra Kübler, et al. 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.

Stanojević, Miloš and Mark Steedman. 2020. Max-margin incremental CCG parsing. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4111–4122. https://doi.org/10.18653/v1/2020.acl-main.378

Tsarfaty, Reut, Joakim Nivre, and Evelina Andersson. 2012. Joint evaluation of morphological segmentation and syntactic parsing. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 6–10.

Vinyals, Oriol, Lukasz Kaiser, Terry Koo, Slav Petrov, Ilya Sutskever, and Geoffrey E. Hinton. 2015. Grammar as a foreign language. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*. pages 2773–2781.

Wei, Yang, Yuanbin Wu, and Man Lan. 2020. A span-based linearization for constituent trees. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 3267–3277. https://doi.org/10.18653/v1/2020.acl-main.299

Yamaki, Ryosuke, Tadahiro Taniguchi, and Daichi Mochihashi. 2023. Holographic CCG parsing. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 262–276. https://doi.org/10.18653/v1/2023.acl-long.15