jp-evalb: Robust Alignment-based PARSEVAL Measures

Jungyeul Park[1] **Junrui Wang**[1] **Eunkyul Leah Jo**[2,3] **Angela Yoonseo Park**[1]
[1]Department of Linguistics, The University of British Columbia, Canada
[2]Department of Computer Science, The University of British Columbia, Canada
[3]Faculté des Sciences et Ingénierie, Sorbonne Université, France
jungyeul@mail.ubc.ca {wjr0710,eunkyul,apark03}@student.ubc.ca

Abstract

We introduce an evaluation system designed to compute PARSEVAL measures, offering a viable alternative to evalb commonly used for constituency parsing evaluation. The widely used evalb script has traditionally been employed for evaluating the accuracy of constituency parsing results, albeit with the requirement for consistent tokenization and sentence boundaries. In contrast, our approach, named jp-evalb, is founded on an alignment method. This method aligns sentences and words when discrepancies arise. It aims to overcome several known issues associated with evalb by utilizing the 'jointly preprocessed (JP)' alignment-based method. We introduce a more flexible and adaptive framework, ultimately contributing to a more accurate assessment of constituency parsing performance.

1 Introduction

For constituency parsing, whether statistical or neural, we rely on the evalb implementation[1], which implements the PARSEVAL measures (Black et al., 1991) as the standard method for evaluating parser performance. There is also a variant of the evalb_spmrl implementation specifically designed for the SPMRL shared task, allowing the evaluation to consider functional phrase labels (Seddah et al., 2013, 2014). 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 non-terminal symbol and span (starting and end indexes). Despite its success in evaluating language technology, evalb faces unresolved critical issues in our discipline. evalb imposes constraints, demanding consistent tokenization and sentence boundary outcomes. Its implementation assumes equal-length gold and system files with matching terminal nodes.

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

In machine translation (MT), sentence alignment involves identifying corresponding sentences in two or more languages and linking sentences from one language to their corresponding counterparts in another. Sentence alignment has been a subject of study for many years, leading to the development of various algorithms. Early research in this area relied on statistical methods that used bilingual corpora to create models capturing the lexical equivalence between words in different languages. For instance, the Gale-Church algorithm, based on sentence length, was one such approach (Gale and Church, 1993). Bleualign introduced a more advanced iterative bootstrapping approach building on length-based methods (Sennrich and Volk, 2011). Earlier approaches also aimed to enhance sentence alignment methodologies by incorporating lexical correspondences, as seen in hunalign (Varga et al., 2005) or the IBM-model based lexicon translation approach (Moore, 2002). Some attempts involved the integration of linguistic knowledge, heuristics, and various scoring methods to improve efficiency, as demonstrated by vecalign (Thompson and Koehn, 2019). Word alignment methodologies are also employed to establish correspondences between words in one language and their direct translations in another. Widely used IBM models (Brown et al., 1993), along with tools like giza++ (Och and Ney, 2000, 2003) or BerkeleyAligner (Liang et al., 2006; DeNero and Klein, 2007), are capable of aligning words.

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. Therefore, 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, constituency parsing maintains the hierarchical structure of a sentence, which

Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 3: System Demonstrations), pages 70–77

can still be valuable for understanding the syntactic relationships between words and phrases. Various studies on formal syntax have focused on constituent structures, such as combinatory categorial grammar (CCG) parsing (Lewis et al., 2016; Lee et al., 2016; Stanojević and Steedman, 2020; Yamaki et al., 2023) or tree-adjoining grammar (TAG) parsing (Kasai et al., 2017, 2018) (whereas CCG and TAG also inherently incorporate dependency structures). In addition, there have been ongoing studies on constituency parsing, such as the linearization parsing method (Vinyals et al., 2015; Liu and Zhang, 2017a,b; Fernández-González and Gómez-Rodríguez, 2020; Wei et al., 2020). If a method that utilizes constituent structures is designed to achieve the goal of creating an end-to-end system, it requires more robust evaluation methods for their constituent structure evaluation.

This paper builds upon our recently introduced alignment-based algorithm, for computing PARSEVAL measures (Jo et al., 2024), which offers a novel approach for calculating precision, recall, and F scores, even in cases of sentence and word mismatch. The primary objective of this paper is to replicate the outcomes generated by `evalb` during the evaluation process. This aims to achieve a comprehensive understanding of the parser's performance by addressing the previous issues of `evalb` and preserving its long-standing legacy. It includes the numbers of gold, test, matched brackets, and cross brackets, as well as precision, recall, and F scores. Furthermore, we present the number of correct POS tags and their tagging accuracy, following a methodology employed by `evalb`. Our proposed method `jp-evalb` is particularly crucial in end-to-end settings, where deviations from the gold file may arise due to variations in tokenization and sentence boundary results.

2 Detailing the `jp-evalb` 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 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.

Algorithm 1 Pseudo-code for `jp-evalb`

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($\mathcal{L}$, $\mathcal{R}$)
  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($\mathcal{T}_{\mathcal{L}'_i(l')}, 0$)
    $\mathcal{S}_{\mathcal{T}_\mathcal{R}} \leftarrow$ GETCONSTITUENT($\mathcal{T}_{\mathcal{R}'_i(r')}, 0$)
    $\mathcal{C}(\mathcal{T}_\mathcal{L}) \leftarrow \mathcal{C}(\mathcal{T}_\mathcal{L})+$ 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 alignment

function ALIGNMENT ($\mathcal{L}$, $\mathcal{R}$):
  while $\mathcal{L}$ and $\mathcal{R}$ **do**
    if Matched CASES $_{(i,j)}$ **then**
      $\mathcal{L}', \mathcal{R}' \leftarrow \mathcal{L}' + \mathcal{L}_i, \mathcal{R}' + \mathcal{R}_j$
    else
      while ¬(Matched CASES $_{(i+1,j+1)}$ **do**
        if LEN($\mathcal{L}_i$) < 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 $\mathcal{L}', \mathcal{R}'$

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 the sentence alignment algorithm. Algorithm 2 shows the generic pattern-matching approach of the alignment algorithm where sentence and word alignment can be applied. We define the following two cases for matched CASES$_{(i,j)}$ of sentence alignment:

$$\mathcal{L}_{i(\not\mid)} = \mathcal{R}_{j(\not\mid)} \quad (1)$$

$$(\mathcal{L}_{i(\not\mid)} \simeq \mathcal{R}_{j(\not\mid)}) \wedge (\mathcal{L}_{i+1(\not\mid)} = \mathcal{R}_{j+1(\not\mid)} \vee \mathcal{L}_{i+1(\not\mid)} \simeq \mathcal{R}_{j+1(\not\mid)}) \quad (2)$$

where we examine whether $\mathcal{L}_i$ is similar to or equal ($\simeq$) to $\mathcal{R}_j$ based on the condition that the ratio of edit distance to the entire character length is less

than 0.1 in (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 token. 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 of $\mathcal{T}_{\mathcal{L}'_i}$ and $\mathcal{T}_{\mathcal{R}'_j}$. The word alignment algorithm adopts a logic similar to sentence alignment. It involves the accumulation of words in l' and r' under the condition that pairs of l_i and r_j do not match, often attributed to tokenization mismatches. Here, we assume interchangeability between notations of sentence alignment ($\mathcal{L}$) and word alignment (l_i). We define the following two cases for matched $\text{CASES}_{(i,j)}$ of word alignment:

$$l_i = r_j \tag{3}$$

$$(l_i \neq r_j) \wedge (l_{i+1} = r_{j+1}) \tag{4}$$

When deciding whether to accumulate the token from l_{i+1} or r_{j+1} in the case of a word mismatch, we base our decision on the following condition, rather than a straightforward comparison between the lengths of the current tokens l_i and r_j: $(\text{LEN}(l) - \text{LEN}(l_{0..i})) > (\text{LEN}(r) - \text{LEN}(r_{0..j}))$

Finally, we extract a set of constituents, 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 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. 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 $\{^{1.0}can\ ^{1.1}not\}$ (Figure 1a).

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	$^{3}right$

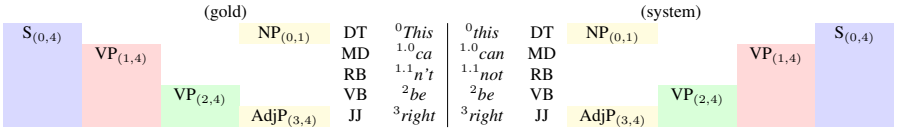
The effectiveness of the word alignment approach remains intact even for morphological mismatches where "morphological segmentation is not the inverse of concatenation" (Tsarfaty et al., 2012), such as in morphologically rich languages. For example, we trace back to the sentence in Hebrew described in Tsarfaty et al. (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'	

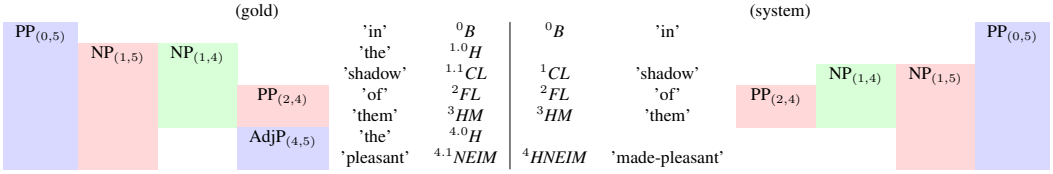
Pairs of $\{^{1.0}H\ ^{1.1}CL, ^{1}CL\}$ ('the shadow') and $\{^{4.0}H\ ^{4.1}NEIM, ^{4}HNEIM\}$ ('the pleasant') are word-aligned using the proposed algorithm, resulting in a precision of 4/4 and recall of 4/6 (Figure 1b).

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 (Figure 1c).

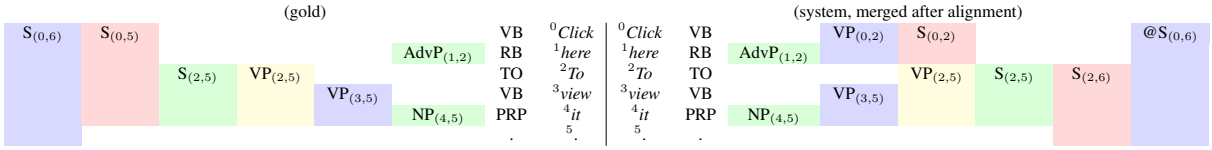
Assumptions To address morphological analysis discrepancies in the parse tree during evaluation, we establish the following two assumptions: (i) The entire tree constituent can be considered a true positive, even if the morphological segmentation or analysis differs from the gold analysis, as long



(a) Example of word mismatches



(b) Example of word mismatches with additional morphemes



(c) Example of sentence mismatches

Figure 1: Example of word and sentence mismatches

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 Usage of `jp-evalb`

We use the following command to execute the `jp-evalb` script:

```
% python3 jp-evalb.py gold_parsed_file \
                      system_parsed_file
```

It generates the same output format as `evalb`. We provide information for each column in both `jp-evalb` and `evalb`, while highlighting their differences. We note that the IDs in `jp-evalb` may not be exactly the same as in `evalb` due to the proposed method performing sentence alignment before evaluation.

`Sent. ID`, `Sent. Len.`, `Stat.` ID, length, and status of the provided sentence, where status 0 indicates 'OK,' status 1 implies 'skip,' and status 2 represents 'error' for `evalb`. We do not assign skip or error statuses.

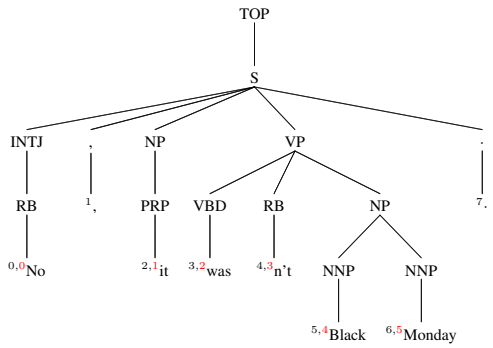
`Recall`, `Precision` Recall and precision of constituents.

`Matched Bracket`, `Bracket gold`, `Bracket test` Assessment of matched brackets (true positives) in both the gold and test parse trees, and their numbers of constituents.

`Cross Bracket` The number of cross brackets.

`Words`, `Correct Tags`, `Tag Accuracy` Evaluation of the number of words, correct POS tags, and POS tagging accuracy.

It's important to note that the original `evalb` excludes problematic symbols and punctuation marks in the tree structure. Our results include all tokens in the given sentence, and bracket numbers reflect the actual constituents in the system and gold parse trees. Accuracy in the last column of the result is determined by comparing the correct number of POS-tagged words to the gold sentence including punctuation marks, differing from the original `evalb` which doesn't consider word counts or correct POS tags. Figure 2 visually depicts the difference in constituent lists between `jp-evalb` and `evalb`. The original `evalb` excludes punctuation marks from its consideration of constituents, resulting in our representation of word index numbers in red for `evalb`. Consequently, `evalb` displays constituents without punctuation marks and calculates POS tagging accuracy based on six word tokens. On the other hand, `jp-evalb` includes punctuation marks in constituents and evaluates POS tagging accuracy using eight tokens, which includes two punctuation marks in the sentence. We note that the inclusion of punctuation marks in the constituents does not affect the total count, as punctuation marks do not constitute a constituent by themselves.



(a) Example of the parse tree

```
('S', 0, 8, "No , it was n't Black Monday .")
('INTJ', 0, 1, 'No')
('NP', 2, 3, 'it')
('VP', 3, 7, "was n't Black Monday")
('NP', 5, 7, 'Black Monday')
```

(b) List of constituents by `jp-evalb`

```
('S', 0, 6, "No it was n't Black Monday")
('INTJ', 0, 1, "No")
('NP', 1, 2, "it")
('VP', 2, 6, "was n't Black Monday")
('NP', 4, 6, "Black Monday")
```

(c) List of constituents by `evalb`

Figure 2: Difference between `jp-evalb` and `evalb`

Additionally, we offer a legacy option, `-evalb`, to precisely replicate `evalb` results. To execute the script with the `evalb` option, utilize the following command:

```
% python3 jp-evalb.py gold_parsed_file \
                      system_parsed_file \
                      -evalb param.prm
```

This option can utilize the default values from the `COLLINS.prm` file if the parameter file is not provided. It will accurately reproduce `evalb` results, even in cases where there are discrepancies such as `Length unmatch` and `Words unmatch` errors in `evalb`'s output. These discrepancies are indicated by the `Stat.` column, which displays either 1 (skip) or 2 (error).

5 Case Studies

Section 23 of the English Penn treebank Under identical conditions where sentences and words match, the proposed method requires around 4.5 seconds for evaluating the section 23 of the Penn Treebank. On the same machine, `evalb` completes the task less than 0.1 seconds. We do not claim that our proposed implementation is fast or faster than `evalb`, recognizing the well-established differences in performance between compiled languages like C, which `evalb` used, and interpreted languages such as Python, which our current implementation uses. Our proposed method also introduces additional runtime for sentence and word alignment, a process not performed by `evalb`. We present excerpts from three result files generated by `evalb` and our proposed method in Figure 3. The parsed results were obtained using the PCFG-LA Berkeley parser (Petrov and Klein, 2007). It's worth noting that there may be slight variations between the two sets of results because `evalb` excludes constituents with specific symbols and punctuation marks during evaluation. However, as we mentioned earlier, `jp-evalb` can reproduce the exact same results as `evalb` for a legacy reason.

Sent ID	L	St	Re	Pr	Mt Br	Br gd	Br te	Cr Br	Co Wd	Co Tg	Tag Acc
1	8	0	100.00	100.00	5	5	5	0	8	7	87.50
2	40	0	70.97	73.33	22	31	30	7	40	40	100.00
3	31	0	95.24	95.24	20	21	21	0	31	31	100.00
4	35	0	90.48	86.36	19	21	22	2	35	35	100.00
5	26	0	86.96	86.96	20	23	23	2	26	25	96.15
.....											

(a) Example of `jp-evalb` results considering punctuation marks during evaluation

Sent ID	L	St	Re	Pr	Mt Br	Br gd	Br te	Cr Br	Co Wd	Co Tg	Tag Acc
1	8	0	100.00	100.00	5	5	5	0	6	5	83.33
2	40	0	70.97	73.33	22	31	30	7	37	37	100.00
3	31	0	95.24	95.24	20	21	21	0	26	26	100.00
4	35	0	90.48	86.36	19	21	22	2	32	32	100.00
5	26	0	86.96	86.96	20	23	23	2	24	23	95.83
.....											

(b) Example of `jp-evalb` results with the legacy option, which produces the exact same results as `evalb`

Sent ID	L	St	Re	Pr	Mt Br	Br gd	Br te	Cr Br	Co Wd	Co Tg	Tag Acc
1	8	0	100.00	100.00	5	5	5	0	6	5	83.33
2	40	0	70.97	73.33	22	31	30	7	37	37	100.00
3	31	0	95.24	95.24	20	21	21	0	26	26	100.00
4	35	0	90.48	86.36	19	21	22	2	32	32	100.00
5	26	0	86.96	86.96	20	23	23	2	24	23	95.83
.....											

(c) Example of the original `evalb` results

Figure 3: Examples of evaluation results on Section 23 of the English Penn treebank

Bug cases identified by `evalb` We evaluate bug cases identified by `evalb`. Figure 4 displays all five identified bug cases, showcasing successful evaluation without any failures. In three instances (sentences 1, 2, and 5), a few symbols are treated as words during POS tagging. This leads to discrepancies in sentence length because `evalb` discards symbols in the gold parse tree during evaluation. Our proposed solution involves not disregarding any problematic labels and including symbols as words during evaluation. This approach implies that POS tagging results are based on the entire token numbers. It is noteworthy that `evalb`'s POS tagging results are rooted in the number of words, excluding symbols. The two remaining cases (sen-

tences 3 and 4) involve actual word mismatches where trace symbols (*-*num*) are inserted into the sentences. Naturally, `evalb` cannot handle these cases due to word mismatches. However, as we explained, our proposed algorithm addresses this issue by performing word alignment after sentence alignment.

Sent ID	L	St	Re	Pr	Mt Br	Br gd	Br te	Cr Br	Cr Wd	Co Tg	Tag Acc
1	37	0	77.27	62.96	17	22	27	5	37	30	81.08
2	21	0	69.23	60.00	9	13	15	2	21	17	80.95
3	47	0	77.78	80.00	28	36	35	4	48	43	89.58
4	26	0	33.33	35.29	6	18	17	8	27	19	70.37
5	44	0	42.31	32.35	11	26	34	17	44	33	75.00

Figure 4: Evaluation results of bug cases by `evalb`

Korean end-to-end parsing evaluation We conduct a comprehensive parsing evaluation for Korean, using system-segmented sequences as input for constituency parsing. These sequences may deviate from the corresponding gold standard sentences and tokens. We utilized the following resources for our parsing evaluation to simulate the end-to-end process: (i) A set of 148 test sentences with 4538 tokens (morphemes) from `BGAA0001` of the Korean Sejong treebank, as detailed in Kim and Park (2022). In the present experiment, all sentences have been merged into a single text block. (ii) POS tagging performed by `sjmorph.model` (Park and Tyers, 2019) for morpheme segmentation.[2] The model's pipeline includes sentence boundary detection and tokenization through morphological analysis, generating an input format for the parser. (iii) A Berkeley parser model for Korean trained on the Korean Sejong treebank (Park et al., 2016).[3]. Figure 5 presents the showcase results of end-to-end Korean constituency parsing. Given our sentence boundary detection and tokenization processes, there is a possibility of encountering sentence and word mismatches during constituency parsing evaluation. The system results show 123 sentences and 4367 morphemes because differences in sentence boundaries and tokenization results. During the evaluation, `jp-evalb` successfully aligns even in the presence of sentence and word mismatches, and subsequently, the results of constituency parsing are assessed.

[2] https://github.com/jungyeul/sjmorph

[3] https://zenodo.org/records/3995084

Sent ID	L	St	Re	Pr	Mt Br	Br gd	Br te	Cr Br	Cr Wd	Co Tg	Tag Acc
1	28	0	85.71	85.71	18	21	21	3	29	26	89.66
2	27	0	91.30	84.00	21	23	25	2	28	25	89.29
3	33	0	88.00	88.00	22	25	25	3	35	31	88.57
4	43	0	72.73	72.73	24	33	33	7	43	40	93.02
5	18	0	69.57	84.21	16	23	19	2	19	12	63.16
.....											

Figure 5: Evaluation results of the end-to-end Korean constituency parsing

6 Previous Work

`tedeval` (Tsarfaty et al., 2012) is built upon the tree edit distance (ADD and DEL) by Bille (2005), incorporating the numbers of nonterminal nodes in the system and gold trees. `conllu_eval`[4] treats tokens and sentences as spans. In case of a mismatch in the span positions between the system and gold files on a character level, the file with a smaller start value will skip to the next token until there is no start value mismatch. Similar processes are applied to evaluating sentence boundaries. For `sparseval` (Roark et al., 2006), a head percolation table (Collins, 1999) identifies head-child relations between terminal nodes and calculates the dependency score. Unfortunately, `sparseval` is currently unavailable. `evalb`, the constituency parsing evaluation metric for nearly thirty years, despite inherent problems, has been widely used.

7 Conclusion

Despite the widespread use and acceptance of the previous PARSEVAL measure as the standard tool for constituency parsing evaluation, it faces a significant limitation by requiring 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 employing 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 original `evalb` 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

[4] https://universaldependencies.org/conll18/conll18_ud_eval.py

new measurement approach offers the opportunity to evaluate constituency parsing within an end-to-end pipeline, addressing discrepancies that may arise during earlier steps, such as tokenization and sentence boundary detection. This enables a more comprehensive evaluation of constituency parsing. All codes and results from the case studies can be accessed at https://github.com/jungyeul/alignment-based-PARSEVAL/.

Acknowledgement

This research is based upon 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

Philip Bille. 2005. A survey on tree edit distance and related problems. *Theoretical Computer Science*, 337(1):217–239.

Ezra Black, Steve Abney, Dan Flickinger, Claudia Gdaniec, Ralph Grishman, Phil Harrison, Donald Hindle, Robert Ingria, Frederick Jelinek, Judith L. Klavans, Mark Liberman, Mitch Marcus, Salim Roukos, Beatrice Santorini, and Tomek Strzalkowski. 1991. A Procedure for Quantitatively Comparing the Syntactic Coverage of English Grammars. In *Speech and Natural Language: Proceedings of a Workshop Held at Pacific Grove, California, February 19-22, 1991*, pages 306–311, Pacific Grove, California. DARPA/ISTO.

Peter F. Brown, Stephen A. Della Pietra, Vincent J. Della Pietra, and Robert L. Mercer. 1993. The Mathematics of Statistical Machine Translation: Parameter Estimation. *Computational Linguistics*, 19(2):263–311.

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

John DeNero and Dan Klein. 2007. Tailoring Word Alignments to Syntactic Machine Translation. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 17–24, Prague, Czech Republic. Association for Computational Linguistics.

Daniel Fernández-González 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, Online. Association for Computational Linguistics.

William A. Gale and Kenneth W. Church. 1993. A Program for Aligning Sentences in Bilingual Corpora. *Computational Linguistics*, 19(1):75–102.

Eunkyul Leah Jo, Angela Yoonseo Park, and Jungyeul Park. 2024. A Novel Alignment-based Approach for PARSEVAL Measures. *Computational Linguistics*, pages 1–10.

Jungo Kasai, 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, Copenhagen, Denmark. Association for Computational Linguistics.

Jungo Kasai, 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, New Orleans, Louisiana. Association for Computational Linguistics.

Mija Kim and Jungyeul Park. 2022. A note on constituent parsing for Korean. *Natural Language Engineering*, 28(2):199–222.

Kenton Lee, 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, Austin, Texas. Association for Computational Linguistics.

Mike Lewis, 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, San Diego, California. Association for Computational Linguistics.

Percy Liang, Ben Taskar, and Dan Klein. 2006. Alignment by Agreement. In *Proceedings of the Human Language Technology Conference of the {NAACL}, Main Conference*, pages 104–111, New York City, USA. Association for Computational Linguistics.

Jiangming Liu and Yue Zhang. 2017a. Encoder-Decoder Shift-Reduce Syntactic Parsing. In *Proceedings of the 15th International Conference on Parsing Technologies*, pages 105–114, Pisa, Italy. Association for Computational Linguistics.

Jiangming Liu and Yue Zhang. 2017b. Shift-Reduce Constituent Parsing with Neural Lookahead Features. *Transactions of the Association for Computational Linguistics*, 5:45–58.

Robert C. Moore. 2002. Fast and Accurate Sentence Alignment of Bilingual Corpora. In *Proceedings of the 5th Conference of the Association for Machine*

Translation in the Americas on Machine Translation: From Research to Real Users, pages 135–244, Tiburon, CA, USA. Springer-Verlag.

Franz Josef Och and Hermann Ney. 2000. Improved Statistical Alignment Models. In *Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics*, pages 440–447, Hong Kong. Association for Computational Linguistics.

Franz Josef Och and Hermann Ney. 2003. A Systematic Comparison of Various Statistical Alignment Models. *Computational Linguistics*, 29(1):19–51.

Jungyeul Park, Jeen-Pyo Hong, and Jeong-Won Cha. 2016. Korean Language Resources for Everyone. In *Proceedings of the 30th Pacific Asia Conference on Language, Information and Computation: Oral Papers (PACLIC 30)*, pages 49–58, Seoul, Korea. Pacific Asia Conference on Language, Information and Computation.

Jungyeul Park and Francis Tyers. 2019. A New Annotation Scheme for the Sejong Part-of-speech Tagged Corpus. In *Proceedings of the 13th Linguistic Annotation Workshop*, pages 195–202, Florence, Italy. Association for Computational Linguistics.

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

Brian Roark, Mary Harper, Eugene Charniak, Bonnie Dorr, Mark Johnson, Jeremy Kahn, Yang Liu, Mari Ostendorf, John Hale, Anna Krasnyanskaya, Matthew Lease, Izhak Shafran, Matthew Snover, Robin Stewart, and Lisa Yung. 2006. SParseval: Evaluation Metrics for Parsing Speech. In *Proceedings of the Fifth International Conference on Language Resources and Evaluation (LREC'06)*, pages 333–338, Genoa, Italy. European Language Resources Association (ELRA).

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. Association for Computational Linguistics.

Rico Sennrich and Martin Volk. 2011. Iterative, MT-based Sentence Alignment of Parallel Texts. In *Proceedings of the 18th Nordic Conference of Computational Linguistics (NODALIDA 2011)*, pages 175–182, Riga, Latvia. Northern European Association for Language Technology (NEALT).

Miloš Stanojević 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, Online. Association for Computational Linguistics.

Brian Thompson and Philipp Koehn. 2019. Vecalign: Improved Sentence Alignment in Linear Time and Space. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 1342–1348, Hong Kong, China. Association for Computational Linguistics.

Reut Tsarfaty, 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, Jeju Island, Korea. Association for Computational Linguistics.

Dániel Varga, Lázló Németh, Péter Halácsy, András Kornai, Viktor Trón, and Viktor Nagy. 2005. Parallel corpora for medium density languages. In *Proceedings of the RANLP (Recent Advances in Natural Language Processing)*, pages 590–596, Borovets, Bulgaria.

Oriol Vinyals, 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. Curran Associates, Inc.

Yang Wei, 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, Online. Association for Computational Linguistics.

Ryosuke Yamaki, 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, Toronto, Canada. Association for Computational Linguistics.