Discontinuous Combinatory Constituency Parsing

Zhousi Chen and **Mamoru Komachi**
Faculty of Systems Design
Tokyo Metropolitan University
6-6 Asahigaoka, Hino, Tokyo 191-0065, Japan
{chen-zhousi@ed., komachi@}tmu.ac.jp

Abstract

We extend a pair of continuous combinator-based constituency parsers (one binary and one multi-branching) into a discontinuous pair. Our parsers iteratively compose constituent vectors from word embeddings without any grammar constraints. Their empirical complexities are subquadratic. Our extension includes 1) a *swap* action for the orientation-based binary model and 2) biaffine attention for the chunker-based multi-branching model. In tests conducted with the Discontinuous Penn Treebank and TIGER Treebank, we achieved state-of-the-art discontinuous accuracy with a significant speed advantage.

1 Introduction

Discontinuity is common in natural languages, as illustrated in Figure 1. Children from a discontinuous constituent are not necessarily consecutive because each can group with its syntactic cousins in the sentence rather than its two adjacent neighbors. Although this relaxation makes discontinuous parsing more challenging than continuous parsing, it becomes more valuable for studies and applications in non-configurational languages (Johnson, 1985), where word order does not determine grammatical function. With gradually saturated continuous parsing accuracy (Zhou and Zhao, 2019; Kitaev and Klein, 2018, 2020; Xin et al., 2021), discontinuous parsing has started gaining more attention (Fernández-González and Gómez-Rodríguez, 2020a, 2021; Corro, 2020).

Typically, constituency parsers are divided into two genres (not including methods employing dependency parsing, e.g., Fernández-González and Gómez-Rodríguez [2020b]): 1) Global parsers use a fixed chart to search through all parsing possibilities for a global optimum. 2) Local parsers rely on fewer local decisions, which leads to lower complexities. The global parser complexities start at least from binary $O(n^3)$ (Kitaev and Klein, 2018) or m-ary $O(n^4)$ (Xin et al., 2021), resulting in low speeds for long parses. Global parsers introduce numerous hypotheses, whereas a local shift-reduce process or incremental parse exhibits more linguistic interests with fewer outputs (Yoshida et al., 2021; Kitaev et al., 2022). Neural global parsers dominate both continuous and discontinuous parsing (Corro, 2020; Ruprecht and Mörbitz, 2021) in terms of F1 score, but they do not exhibit a strong accuracy advantage over other parsers. Although local parsers may in some cases produce ill-formed trees, global parsers do not guarantee the selection of gold-standard answers from their charts.

We propose extending a pair of local parsers to achieve high speed, accuracy, and convenient investigation for both binary and multi-branching parses. Chen et al. (2021) proposed a pair of continuous parsers employing bottom-up vector compositionality. We dub these as neural combinatory constituency parsers (**NCCP**) with binary **CB** and multi-branching **CM**. To the best of our knowledge, they possess the top parsing speeds for continuous constituency. **CB** reflects a linguistic branching tendency, whereas **CM** represents unsupervised grammatical headedness through composition weight. **CM** is among the few parsers that do not require preprocessing binarization (Xin et al., 2021) but possess a high parsing speed. We dub our extension as **DCCP** with binary **DB** and multi-branching **DM**. The mechanisms with neural discontinuous combinators are shown in Figure 2.

Specifically, our combinators take a sentence-representing vector sequence as input and predict layers of concurrent tree-constructing actions. Two mechanisms are employed. **DB** triggers an action if the orientations of two neighboring vectors agree. A *swap* action exchanges the vectors;

Transactions of the Association for Computational Linguistics, vol. 11, pp. 267–283, 2023. https://doi.org/10.1162/tacl_a_00546
Action Editor: Carlos Gómez-Rodríguez. Submission batch: 8/2022; Revision batch: 12/8/2022; Published 3/2023.

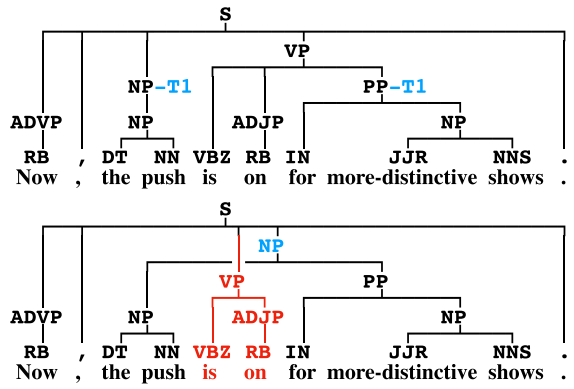


Figure 1: Evang and Kallmeyer (2011, DPTB) recover discontinuity from continuous Penn Treebank (Marcus et al., 1993, PTB) with trace nodes (blue).

a *joint* action composes a new vector with them. **CB** only possesses a *joint* action. Meanwhile, **DM** takes discontinuous vectors to form biaffine attention matrices and decides their groups collectively; the remaining continuous vectors resort to chunking decisions, as with **CM**. **NCCP** and **DCCP** are unlexicalized supervised greedy parsers.

The contributions of our study are as follows:

- We propose a pair of discontinuous parsers[1] (i.e., binary and multi-branching) by extending continuous parsers of Chen et al. (2021).
- We demonstrate the effectiveness of our work on the Discontinuous Penn Treebank (Evang and Kallmeyer, 2011, DPTB) and the TIGER Treebank (Brants et al., 2004). Our parsers achieve new state-of-the-art discontinuous F1 scores and parsing speeds with a small set of training and inferring tricks, including discontinuity as data augmentation, unsupervised headedness, automatic hyperparameter tuning, and pre-trained language models.

2 Related Work

Global Parsing. The chart for binary continuous parsing (Kitaev and Klein, 2018) is triangular, as shown in black in Figure 3. The horizontal dimension enumerates the position of each node (i.e., each bit as a word). The vertical `C` indicates the number of continuous cases included in each node. Nodes at height h share combinatorics of complexity h. Consequently, a binary continuous chart parser has a fixed $\Sigma_{h=0}^{n}(n-h)\cdot h \sim O(n^3)$ complexity for the CKY decoding algorithm.

In m-ary and/or discontinuous cases (i.e., for multi-branching arity m and/or fan-out k of each constituent), the chart is superior to that of binary continuous parsing in terms of complexity. Both horizontal and vertical axes expand to diversify the combination of discrete bits of each lexical node. Because of the expansion, m-ary continuous global parsing (Xin et al., 2021) has $O(n^4)$ complexity, whereas binary discontinuous parsing has exponential complexity at $O(n^{3k})$ (Corro, 2020; Stanojevic and Steedman, 2020), where $k \in \{1, 2\}$ are special cases for binary CFG (likely in Chomsky Normal Form [CNF]) and binary Linear Context-Free Rewriting Systems with maximum fan-out 2 (Stanojevic and Steedman, 2020, LCFRS-2). M-ary discontinuous parsing, which certainly has a higher complexity, is not yet available for global parsing.

For efficiency, expensive rules are commonly restricted by LCFRS-2 parsers (Corro, 2020; Stanojevic and Steedman, 2020; Ruprecht and Mörbitz, 2021). A tricky $O(n^3)$ variant of Corro (2020) covering major rules has produced the best results. However, the variant excludes 2% sophisticated discontinuous rules on TIGER Treebank. Limited by the simplified grammar, their discontinuous scores are low, especially for recalls. A global optimum does not guarantee a gold parse, leaving room for local parsing.

Local Parsing. Local parsers do not observe the chart framework and only consider one greedy or a few hypotheses. **Transition-based** parsers with a *swap* or *gap* action have sequential actions and low complexities (Maier, 2015; Coavoux and Crabbé, 2017). Multiple *swaps* or *gaps* combine to construct a large discontinuous constituent. In contrast, **stack-free** parsing can directly pick up a distant component with one attachment search (Coavoux and Cohen, 2019). **Easy-first** (Nivre et al., 2009; Versley, 2014) and **chunker-based** (Ratnaparkhi, 1997; Collobert, 2011) run rapidly.

Fernández-González and Gómez-Rodríguez (2020a) redirected discontinuity to **dependency** parsing via pointer networks and obtained significant accuracy improvement among greedy parsers. However, it is difficult to determine whether such improvement originates from the model or the extra head information. In contrast, Fernández-

[1] Our code with all model configuration files is available at `https://github.com/tmu-nlp/UniTP`.

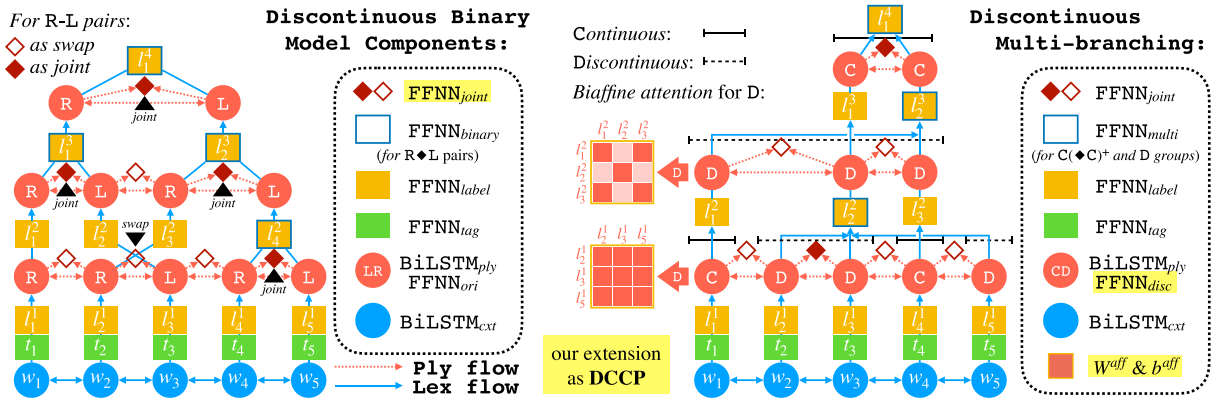


Figure 2: Mechanism examples. Left: **DB** produces *swap* to facilitate traveling of discontinuous nodes and *joint* to combine adjacent nodes. Right: **DM** leverages biaffine attention to identify and combine discontinuous groups.

Layer & Fan-out		Bits	#×C	#×D
#3	$k=1$	1111	1×3	1×7
#2	$k=2$	1101 1011		4×3
	$k=1$	1110 0111	2×2	
#1	$k=2$	1010 1001 0101		6×1
	$k=1$	1100 0110 0011	3×1	
#0	$k=1$	1000 0100 0010 0001	0	0

Figure 3: Fan-out and input sizes increase Continuous and Discontinuous combinatory global parser cases.

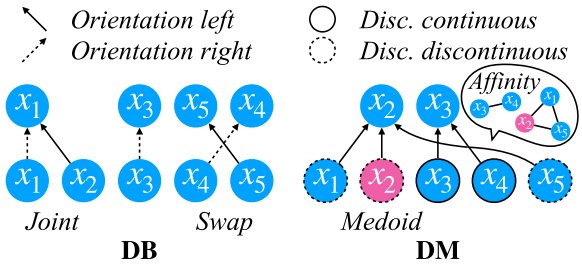


Figure 4: **DCCP** plies with concurrent actions.

González and Gómez-Rodríguez (2020b) **reorder**ed input words in at most $O(n^2)$ complexity and redirected to various continuous parsers.

3 Discontinuous Combinatory Parsing

We call a level of partial derivations or subtrees a **ply** (Jurafsky and Martin, 2009), as depicted in Figure 4. Similar to the state of a transition-based parser, each of which leads to an action, ply is the state for **DCCP**, each of which leads to a sequence of concurrent actions for itself.

Starting with a sequence of words $(x_1, \cdots, x_n)$ as an initial ply, we assemble an unlabeled discontinuous parse tree in a bottom–up manner by applying concurrent actions to the roots of subtrees in the ply and iterating until sequence length $n = 1$.

3.1 Binary Ply: *Joint* and *Swap*

For a binary tree, two actions are sufficient:

$$\begin{aligned} &\mathit{action}(x_i \oplus x_{i+1}) \in \{\mathit{joint},\ \mathit{swap}\} \\ &\mathit{joint} : \mathit{compose}(x_i,\ x_{i+1}) \rightarrow x_i \\ &\mathit{swap} : (x_i,\ x_{i+1}) \rightarrow (x_{i+1},\ x_i)\,. \end{aligned} \tag{1}$$

One *joint* reduces the sequence length by one; the *swap* does not affect the sequence length but affects its order. The binary function *compose* is a binary neural combinator. The concatenation "$\oplus$" only works for adjacent nodes.

However, concurrent adjacent actions would conflict in a ply (e.g., two *swap*s for (x_1, x_2, x_3) leaves an undecidable x_2). In other words, they need a resolution. We adopt a solution of

$$\begin{aligned} &\mathit{orientation}(x_i) \in \{0,\ 1\} \\ &\mathit{orientation}(x_i) - \mathit{orientation}(x_{i+1}) = 1\,, \end{aligned} \tag{2}$$

where each *orientation* indicates either *left* (0) or *right* (1) and the adjacent node pair of $(x_i,\ x_{i+1})$ contains the agreeing orientations. Thus, only under this circumstance, **DB** activates *joint* or *swap* by $\mathit{action}(x_i \oplus x_{i+1})$ without conflict.

Summary. All nodes in a **DB** ply are derived by Formula 1 under the condition of Formula 2 to form a new ply. As exemplified in Figure 4 for **DB**, $(x_1,\ x_2)$ and $(x_4,\ x_5)$ meet the condition of Formula 2 and have respective *joint* and *swap* actions. Meanwhile, x_3 takes neither action and remains in the ply, because the *orientations* of x_2

and x_4 do not agree with x_3, regardless of x_3's *orientation*.

3.2 Multi-branching Ply: *Affinity* and *Chunk*

We characterize whether x_i and x_j from a ply are two siblings of the same parent constituent as

$$\textit{affinity}(x_i, x_j) \in \{0,\ 1\}\,, \tag{3}$$

where 0 denotes false and 1 denotes true. Thus, **DM** decides a *discontinuity* action of x_i and then forwards it to a group action for either a *discontinuous* or *continuous* constituent as in Formula 4:

$$\begin{aligned}
&\textit{action}(x_i) \in \{\textit{discontinuous},\ \textit{continuous}\}\\
&\textit{discontinuous}:\\
&\quad \mathcal{G} = \{x_j \mid \textit{affinity}(x_i,\ x_j) = 1\}\\
&\quad\quad \textit{medoid} \in \{j \mid x_j \in \mathcal{G}\}\\
&\quad\quad \textit{compose}(\mathcal{G}) \rightarrow x_{\textit{medoid}}\\
&\textit{continuous}:\\
&\quad \mathcal{G} = \{x_j \mid \textit{lb} < i \leq \textit{rb},\ \textit{lb} < j \leq \textit{rb},\ \text{and}\\
&\quad\quad \textit{affinity}(x_j,\ x_{j+1}) = \mathbb{1}(j \notin \{\textit{lb},\ \textit{rb}\})\}\\
&\quad\quad \textit{compose}(\mathcal{G}) \rightarrow x_{\textit{lb}+1}\,,
\end{aligned} \tag{4}$$

whereas ‘‘$\mathbb{1}(\cdot)$’’ is the indicator function. We select one *medoid* for each *discontinuous* constituent to determine its position in the modified ply, whereas the choice of *medoid* for *continuous* constituents makes no difference. *Continuous* nodes split into segments of $(x_{lb+1},\ \cdots,\ x_{rb})$ with $(lb,\ lb+1)$ and $(rb,\ rb+1)$ as boundaries. Function *compose* is a flexible m-ary neural combinator with $m \in \mathbb{N}$.

Dozat and Manning (2017) characterized each dependency tree as a sparse asymmetric matrix via biaffine attention, with each sole positive signal in a row (or column) indicating a lexical dependency (from a word to its head or vice versa). Nevertheless, lexical dependency is not available for constituency parsing, and biaffine attention becomes expensive at $O(n^2)$ complexity.

In contrast, we designate *discontinuous affinity* as a small dense symmetric biaffine attention matrix and control its computational size of $O(n^2)$. Otherwise, *continuous affinity* for adjacent nodes takes a special form of

$$\textit{chunk}(x_i \oplus x_{i+1}) = \textit{affinity}(x_i, x_{i+1})$$

with a simpler $O(n)$ complexity.

Summary. Via fast chunking or a small biaffine attention matrix, **DM** balances to increase its efficiency. As exemplified in Figure 4 for **DM**, *discontinuous* (x_1, x_2, x_5) are grouped as one because of their mutual *affinity*, which is equivalent to a 3×3 biaffine attention matrix of ones. Node x_2 is selected as the *medoid* for the constituent's location in the new ply. Meanwhile, *continuous* (x_3, x_4) forms a constituent for $\textit{chunk}(x_i \oplus x_{i+1}) = \textit{affinity}(x_i,\ x_{i+1}) = \mathbb{1}(i \notin \{2, 4\})$ and $i \in [2, 4]$.

3.3 Oracle

We state the conversion from tree into layers of action signals for fully supervised training. For convenience, we merge **DM**'s *chunk* semantics into *joint* to unify **DB**'s and **DM**'s interstice signals. The common signals are

$$\left(x_{1:n},\ t_{1:n},\ l^{1:H}_{1:n_h},\ j^{1:H-1}_{1:n_h-1}\right),$$

where x represents a sentence with n words, n POS *tags*, H layers of *labels*, and $H-1$ layers of *joints*, respectively. ‘‘:’’ indicates sequence range (e.g., ply height $h \in [1, H]$). **DB** features in

$$\left(o^{1:H-1}_{1:n_h}\right),$$

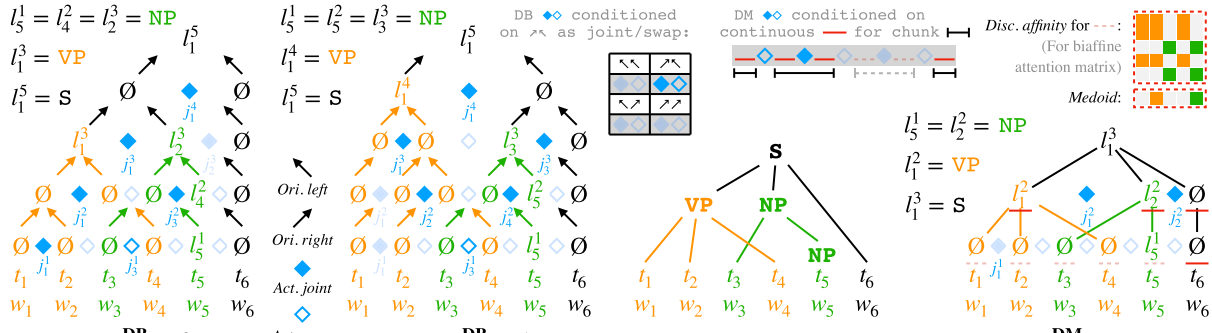
containing $H-1$ layers of *orientations*. For **DM**, our extension includes *discontinuity* and *affinity* biaffine attention matrices with *medoids*,

$$\left(d^{1:H-1}_{1:n_h},\ a^{1:H-1}_{[1:d_h,\,1:d_h]}\right),$$

where $d_h = \sum_{i=1}^{n_h} d^h_i \leq n_h$ indicates a number of discontinuous nodes that is no larger than the number of total nodes in layer h.

Empty Node and Unary Branch. Similar to a range of previous works (Chen et al., 2021; Shen et al., 2018; Kitaev and Klein, 2020; Corro, 2020), we adopt an empty label ‘‘$\varnothing$’’ for our substructure (e.g., binarization). Additionally, we collapse each unary branch into a single node and join their constituent labels according to their hierarchical order (e.g., `S+VP` as derivation `S`→`VP`) with easy restoration during inference. Unary collapse is productive for label type, as shown in Table 1.

Binarization $\rho^{\textbf{DB}}$. C children of a constituent join one by one via their *orientation* and *joint* signals. For $c \in [1,\ C)$, $[1,\ c]$-th children are set to *orientation right* (1) and $(c,\ C]$-th are set to



(a) Binary $\rho^{\mathbf{DB}} = 0$ "*left*" (b) Binary $\rho^{\mathbf{DB}} = 1$ "*right*" (c) Original (d) Multi-branching $\rho^{\mathbf{DM}}$ = "*random*"

Figure 5: Examples illustrating the principle of stratification. The original m-ary tree (c) is binarized and stratified into (a) and (b) with numeric factors $\rho^{\mathbf{DB}}$, whereas (c) is stratified into (d) with a categorical *medoid* factor *random*. In (d), w_2 and w_5 are randomly selected as *medoids* for discontinuous parents l_1^2 and l_2^2 with more or less twisted descendant lines. We color constituent components and show disabled *joints* with light blue "◆" and "◇."

Corpus	Label Type Total	Label Type Collapsed	Label Token Frequency%
DPTB	126	99	3.82%
TIGER	80	55	0.72%

Table 1: Label type and token in our oracle format.

orientation left (0). Neighboring children have positive *joint* signals if they are siblings. Otherwise, negative *joints* swap them toward their siblings. We normalize a factor $\rho^{\mathbf{DB}} = \frac{c-1}{C-2} \in [0, 1]$ for treebank binarization. In continuous parsing, $\rho^{\mathbf{DB}} \in \{0, 1\}$ implies CNF.

As illustrated in Figure 5 **(a)** & **(b)**, we obtain layers of action signals from the binarization of (c). As an extension to CNF, we use the beta distribution $\rho^{\mathbf{DB}} \sim \text{Beta}(\alpha_{left}, \alpha_{right}) \in (0, 1)$ to create augmented samples with $\alpha_{left}, \alpha_{right} \in (0, +\infty)$.

Medoid $\rho^{\mathbf{DM}}$. We use a set of categorical *medoid* factors $\rho^{\mathbf{DM}} \in \{random, leftmost, rightmost\}$ to stratify a multi-branching tree: 1) *random* picks a random child with uniform probability, whereas 2) *leftmost* and 3) *rightmost* take the two ends of a discontinuous group.

In Figure 5 **(d)**, w_1 and w_4 are randomly selected as *medoids*. Meanwhile, l_2^1 and l_2^2 would exchange their places if w_3 and w_4 were selected. *Medoid* is different from headedness (Zwicky, 1985). It is an intermediate variable for locating a discontinuous constituent.

3.4 Model Implementation

NCCP and **DCCP** have the same bottom–up iteration on a *ply* and share two types of neural components: bidirectional Long Short-Term Memory (BiLSTM) and feedforward neural network (FFNN).

Algorithm 1: Combinatory Parsing

Function PARSE$(x_{1:n})$**:**
  ply $\leftarrow$ [] and $n_h \leftarrow n$ with height $h \leftarrow 1$;
  $x^h_{1:n_h} \leftarrow \text{BiLSTM}_{cxt}$ (*embeddings of* $x_{1:n}$);
  for $i \leftarrow 1$ **to** n **do**
    $\hat{t}_i \leftarrow \text{FFNN}_{tag}(x_i^h)$;
    $\hat{l}_i^h \leftarrow \text{FFNN}_{label}(x_i^h)$;
    append a tree with $(x_i, \hat{t}_i, \hat{l}_i^h)$ to *ply*;
  while $n_h > 1$ **do**
    $z^h_{1:n_h} \leftarrow \text{BiLSTM}_{ply}(x^h_{1:n_h})$;
    $x^{h+1}_{1:n_{h+1}} \leftarrow \text{FOLD}(ply, x^h_{1:n_h}, z^h_{1:n_h})$;
    $h \leftarrow h + 1$;
    for $i \leftarrow 1$ **to** n_h **do**
      $\hat{l}_i^h \leftarrow \text{FFNN}_{label}(x_i^h)$;
      label i-th tree of *ply* with $\hat{l}_i^h$;
  return the sole tree in *ply*;

In Algorithm 1, BiLSTM_{cxt} contextualizes the sequence of words $x_{1:n}$ as embeddings $x^1_{1:n_1}$ and BiLSTM_{ply} contextualizes the ply sequence $x^h_{1:n_h}$ for either **DB** FOLD in Algorithm 2 or **DM** FOLD in Algorithm 3, either of which modifies *ply* and constructs new layers of embeddings $x^{h+1}_{1:n_{h+1}}$. The necessity of BiLSTM_{cxt} and BiLSTM_{ply} contextualization was empirically examined with **NCCP**. Meanwhile, FFNN_{tag} and FFNN_{label} predict the lexical tags and constituent labels based on contextualized individual embeddings without grammar constraint.

When actions do not modify *ply* at inference phase, PARSE terminates with a VROOT label for the current *ply*. We define CONDENSE as a

Algorithm 2: Binary Ply

Function FOLD(*ply*, $x_{1:n}$, $z_{1:n}$):
  for $i \leftarrow 1$ **to** n **do**
    $\hat{o}_i^h \leftarrow \text{FFNN}_{ori}(z_i)$;
    $\hat{j}_i^h \leftarrow \text{FFNN}_{joint}(z_i \oplus z_{i+1})$;
    w.r.t. Subsection 3.1, apply actions to *ply* and $x_i \leftarrow \text{COMPOSE}(x_i, x_{i+1})$;
  return CONDENSE($x_{1:n}$);
Function COMPOSE(x_L, x_R):
  $\lambda \leftarrow \sigma\ \text{FFNN}_{binary}(x_L \oplus x_R)$;
  return $\lambda \odot x_L + (1-\lambda) \odot x_R$;

process reenumerating ply nodes regardless their inconsecutive and unordered indices following the actions (e.g., $\text{CONDENSE}(x_1, x_3, x_5, x_4) \rightarrow (x_1, x_2, x_3, x_4)$)

$$\text{CONDENSE} : (n \text{ ply nodes}) \rightarrow (x_1, \ldots, x_n)$$

Binary Combinator. The FOLD of **DB** is shown in Algorithm 2. The COMPOSE function uses sigmoid activation "σ" to create a pair of complementary gates λ and $(1-\lambda)$ for x_L and x_R. λ is a vector of the same size as the embeddings.

Multi-branching Combinator. The **DM** FOLD has vectors $z_{1:n}$ and $\Delta_{1:n}$ in exchangeable shapes in Algorithm 3. We choose $\Delta_{1:n}$ because it performs empirically optimal. Otherwise, Algorithms 2 and 3 should look more identical. Meanwhile, λ_i in COMPOSE with $\sum_i^{\mathcal{G}} \lambda_i = \mathbf{1}$ from *Softmax* is the adaptive gating vector for x_i. We consider the average of λ_i (i.e., $\bar{\lambda}_i$) as the *unsupervised headedness* for inference and visualization. Thus, **DM** is able to *infer* with a special factor:

$$\rho^{\mathbf{DM}} = \mathit{uhead}\,,$$

which takes $\mathit{medoid} \leftarrow \arg\max_{i \in \mathcal{G}} \bar{\lambda}_i$ as the group medoid.

To identify discontinuous groups in the *affinity* biaffine attention matrix, **DM** takes $M = \sigma \hat{a}^h_{[1:D,1:D]}$ for value range $(0,1)$ ($D = \sum_i \hat{d}_i^h$) and booleanizes it into $B \leftarrow M > \theta$. It 1) tries default threshold $\theta = 0.5$ as the natural selection for sigmoid activation and **check**s whether all the following statements are true:

- B is symmetric (i.e., $B = B^\intercal$),
- any rows $v, w \in B$ are $v \neq 0$,
- either $v = w$ or $v^\intercal \cdot w = 0$.

Algorithm 3: Multi-branching Ply

Function FOLD(*ply*, $x_{1:n}$, $z_{1:n}$):
  $(\vec{z}_{1:n} \oplus \overleftarrow{z}_{1:n}) \leftarrow z_{1:n}$ (forward & backward);
  for $i \leftarrow 1$ **to** n **do**
    $\hat{d}_i^h \leftarrow \text{FFNN}_{disc}(z_i)$;
    $\Delta_i \leftarrow (\vec{z}_i - \vec{z}_{i-1}) \oplus (\overleftarrow{z}_i - \overleftarrow{z}_{i+1})$;
    $\hat{j}_i^h \leftarrow \text{FFNN}_{joint}(\Delta_i \oplus \Delta_{i+1})$;
  foreach i,j-th *discontinuous* $\hat{d}^h_{v,w}$ **do**
    $\hat{a}^h_{[i,j]} \leftarrow x_v^\intercal \cdot W^{aff} \cdot \Delta_w + b^{aff}$;
  find *discontinuous* groups by **check**ing $\hat{a}^h$;
  foreach group indices $\mathcal{G}$ and its *medoid* **do**
    w.r.t Subsection 3.2, apply actions to *ply* and $x_{\mathit{medoid}} \leftarrow \text{COMPOSE}(\Delta_{\mathcal{G}}, x_{\mathcal{G}})$;
  return CONDENSE($x_{1:n}$);
Function COMPOSE($\Delta_{\mathcal{G}}$, $x_{\mathcal{G}}$):
  $\lambda_{\mathcal{G}} \leftarrow \mathit{Softmax}(\text{FFNN}_{multi}(\Delta_{\mathcal{G}}))$;
  return $\sum_i^{\mathcal{G}} \lambda_i \odot x_i$;

It succeeds in most cases. Otherwise, it 2) tries a value from M as θ, **check**s and loops again. We order the thresholds by their distances to the default 0.5. If all 2) fail, it 3) simply falls back into grouping all nodes as one and counts one FAIL.

Basic Losses. We choose HINGE-LOSS for binary prediction and CROSS-ENTROPY for multi-class prediction, following **NCCP**. Respecting the context in Algorithms 1–3, our basic loss items are

$$L_{tag},\ L_{label},\ L_{ori},\ L_{jnt},\ L_{disc},\ L^D_{aff}$$

by accumulating their items across all layers. For example, $L^D_{aff} \leftarrow \sum_{i,j,h} \text{HINGE-LOSS}(a^h_{[i,j]}, \hat{a}^h_{[i,j]})$ with D for *discontinuous affinity*. We have additional loss items in the next subsection.

Complexity. Extreme cases provide the upper bounds for our theoretical complexity. **DB** takes $\frac{n}{2}$ fully swapping plies and $\frac{n}{2}$ fully joining plies. Each ply costs $O(n)$ recurrency; the bound is $O(n^2)$. Every **DM** ply involves a matrix for all nodes and decreases n only by one. Assume that we limit **check** 2) to some fixed sizes. Each ply costs $O(n^2)$; the bound is $O(n^3)$.

However, **DCCP** has an empirical $O(n^2)$ complexity with strong linearity, as shown in Figure 6. **DB** has higher linear coefficients because of its slow binary combination. Meanwhile, **DM** shows stronger quadratic tendency because of biaffine

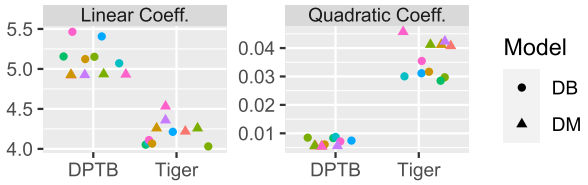


Figure 6: Quadratic linear regression (LR) on sentence length vs. parsing node count on stratified **DCCP** treebanks. **DM** counts in biaffine attention matrix nodes. Colors show binarization and medoid strategies. Cubic LR gives all negative cubic terms highly close to zero.

attention. Yet, their coefficient magnitudes are on par with one another.

3.5 Training Tricks

Data Augmentation. Figure 7 first summarizes **(e)** basic data augmentation with binarization for **DB** and *medoid* for **DM** (including **(a)**, **(b)**, and **(d)** in Figure 5). The beta distribution can resemble a uniform *random* distribution or other biased distributions to detect linguistic branching tendency with specific $(\alpha_{left}, \alpha_{right})$.

Then, we further leverage the intermediate nonterminal node ''$\varnothing$'' to create more $\varnothing$-subtrees. The augmentation is an inspiration of **CM**'s deterministic `_SUB` node, which balances subtree heights and boosts both accuracy and efficiency. (Non-`_SUB` trees remain at their original heights.) However, **(g)** $\varnothing$-subtree is random and creates imbalance. It creates only one stretching branch by iteratively grouping nodes with possibility $\rho_{\varnothing}$, which has three significant impacts:

- Random stretching branches add mild variations to the context as states for robust ply actions in `FOLD`.
- Random discontinuity creates **DB** orientation layers that cannot be created by $\rho^{\mathbf{DB}}$ binarization.
- They reduce large (possibly continuous) constituents into smaller (possibly discontinuous) pieces without adding a large payload to the biaffine attention, which narrows the gap between **DB** and **DM** (**DM** is more vulnerable to dramatic many-to-one `COMPOSE`).

Taking `NP` ''a good day'' for instance, any of ''a day,'' ''a good,'' and ''good day'' can be an intermediate option for creating the `NP`. On the one hand, these options create varied contexts for the remaining parts of a ply. On the other hand, assume that ''a day'' (which is not a $\rho^{\mathbf{DB}}$ product) is selected. **DM** learns to discern it with the other possible ''a day'' in biaffine attention based on their context.

Model Robustness. To further randomize **DB** training, we introduce **(f)** *ply shuffle* and its resultant losses L^{shfl}_{ori} and L^{shfl}_{jnt}. It shuffles $x^{h}_{1:n_h}$ with respect to each constituent, takes the new sequence to $\texttt{BiLSTM}_{ply}$ and `FOLD`, and reuses the ply of *orientation* and *joint* for those additional losses. For example, a `VP` to the left of an `NP` gets shuffled to the right with the same ply ''*right* ♦ *left*'' producing the additional loss items.

Continuous affinity and *discontinuous affinity* in **DM** undergo different identification processes. To minimize the difference, we introduce **(h)** L^{C}_{aff} and L^{X}_{aff} for *continuous* and interply *affinity*, in addition to cardinal L^{D}_{aff}. These reduce the risk of biaffine attention forwarding incorrect nodes, which would evoke exposure bias. We use positive rates β_c and β_x for $\beta_c \cdot \sigma(\hat{d}^{h}_{i})$ and $\beta_x \cdot \sigma(x^{h\intercal}_{w} \cdot W^{aff} \cdot \Delta^{\bar{h}}_{w} + b^{aff})$ to limit the sample size, where layers $\bar{h} \neq h$ contain *discontinuous* nodes. Fallible signals are more likely to form losses via `HINGE-LOSS`.

In summary, the additional loss items are

$$L^{shfl}_{ori},\ L^{shfl}_{jnt},\ L^{C}_{aff},\ L^{X}_{aff}.$$

4 Experiment

DCCP takes frozen pre-trained FastText as static word embedding (PWE) or fine-tuned 12-layer pre-trained XLNet and BERT as contextualized embedding of pre-trained language models (PLM) as lexical input [2] and parses on English DPTB and German TIGER treebanks. See Table 2.

Two-stage Training for a PWE Model. The first stage (**S1**) requires approximately 300 epochs

[2]To compare to other parsers with a lexical component, **NCCP** used pre-trained FastText or FastText trained on PTB. The former slightly increased F1 score by 0.2. We choose BERT (`https://www.deepset.ai/german-bert`) for German and adopt $\texttt{FFNN}_{cxt}$ instead of $\texttt{BiLSTM}_{cxt}$ for model connection following **NCCP**.

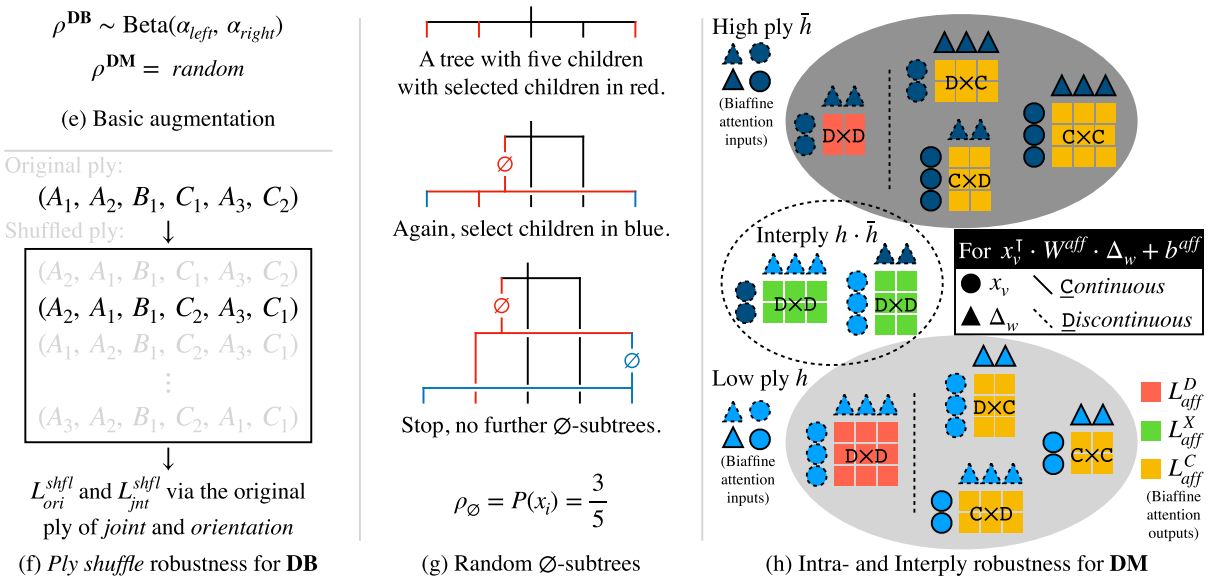


Figure 7: (e) Beta distribution is equivalent to uniform *random* when $\alpha_{left} = \alpha_{right} = 1$. Otherwise, it is for detecting branching tendency with **DB**. (f) Constituent children get shuffled and create additional losses. (A_1, A_2, A_3), B_1, and (C_1, C_2) belong to three different constituents. (g) ∅-subtree creates discontinuity from continuity with a random stretching branch. (h) In-ply continuous nodes and interply nodes are chosen for **DM** biaffine attention.

DCCP model dimension	300
$\texttt{BiLSTM}_{ctx}$ / $\texttt{BiLSTM}_{ply}$ layers	6 / 2
$\texttt{FFNN}_{\{tag, label, ori, jnt, disc\}}$ layers	2
Optimizer	Adam
(1-epoch γ warm-up, linear decay, and early stop.)	
Drop out rate (recurrent)	0.4 (0.2)
Batch size (non-training)	80 (160)

Parameter sizes (w/o PLM) $\texttt{BiLSTM}_{cxt}$	+CB	+CM	+DB	+DM
3.25M	0.36M	0.55M	1.32M	1.45M

Table 2: Fixed model hyperparameters and model parameter sizes of **NCCP** and **DCCP**. **CB** & **CM** have a single-layer $\texttt{BiLSTM}_{ply}$ of fewer model parameters.

with general hyperparameters. Loss functions are sum of all loss items:

$$L^{\mathrm{S1}}_{\mathrm{DB}} = L_{tag} + L_{label} + L_{jnt} + L_{ori} + L^{shfl}_{ori} + L^{shfl}_{jnt}$$
$$L^{\mathrm{S1}}_{\mathrm{DM}} = L_{tag} + L_{label} + L_{jnt} + L_{disc} + \sum_{i \in D,X,C} L^i_{aff}$$

Adam optimizer's learning rate is $\gamma = 10^{-3}$. **DB** uses uniform binarization $\alpha_{left} = \alpha_{right} = 1$. **DM** uses ∅-subtree ratio $\rho_\varnothing = 0.25$, robustness $\beta_c = 0.1$, and $\beta_x = 1$ for both efficiency and accuracy.

The second stage (**S2**) involves 100 short trials with a Bayesian optimization (BO) tool (Akiba et al., 2019, *optuna*); each trial requires less than 30 epochs and brings hyperparameter adjustment:

$$L^{\mathrm{S2}}_{\mathrm{DB}} = \alpha_{tag} \cdot L_{tag} + \cdots + \alpha^{shfl}_{jnt} \cdot L^{shfl}_{jnt}$$
$$L^{\mathrm{S2}}_{\mathrm{DM}} = \beta_{tag} \cdot L_{tag} + \cdots + \sum_{i \in D,X,C} (\beta_i \cdot L^i_{aff}).$$

Trials follow practical constraints: learning rate $\gamma \in (10^{-6}, 10^{-3})$, beta's $\alpha_{left}, \alpha_{right} \in (10^{-3}, 10^{3})$ instead of $(0, +\infty)$, and $[0, 1]$ for the others.

PLM models also use general hyperparameters with learning rate 10^{-6} at **S1**. PLMs are frozen during the first 50 epochs to avoid noise pollution and then are fine-tuned with learning rate 3×10^{-6}. They inherit explored hyperparameters from PWE models at **S2**, except for learning rate 3×10^{-6}.

4.1 Overall Results

Table 3 shows F1 scores of recent neural discontinuous parsers under comparable conditions on test sets. We follow their reported number of significant digits and reduce the effects of random initialization with an average of five runs. The details are shown in Table 4.

DCCP models achieved state-of-the-art performance in terms of discontinuous F1 scores and parsing speeds. Although speed tests are

Model			DPTB test set			TIGER test set		
without pre-trained language model	**Type**	**Complexity**	**F1**	**D.F1**	**Speed**	**F1**	**D.F1**	**Speed**
Coavoux et al. (2019)	Trans-Gap	$O(n)$	91.0	71.3	80	82.7	55.9	126
Coavoux and Cohen (2019)	Stack-Free	$O(n^2)$	90.9	67.3	38	82.5	55.9	64
Pointer-based VG20 w/ Ling et al. (2015)	Seq-Labeling	$O(n^2)$	88.8	45.8	611	77.5	39.5	568
Pointer-based FG22 w/ Ling et al. (2015)	Multitask†	$O(n^2)$	–	–	–	**86.6**	**62.6**	–
Stanojevic and Steedman (2020)	Chart	$O(n^6)$	90.5	67.1	–	83.4	53.5	–
Corro (2020)	Chart	$O(n^3)$	**92.9**	64.9	355	85.2	51.2	474
Ruprecht and Mörbitz (2021) w/ flair	Chart	–	91.8	76.1	86	85.1	61.0	80
DB w/ FastText (en & de)	Combinator	$O(n^2)$	92.0	75.6	**940**	84.9	60.1	**1160**
DM w/ FastText (en & de)	Combinator	$O(n^3)$	92.1	**78.1**	**970**	85.1	62.0	**1300**
with pre-trained language model								
Pointer-based VG20 w/ $\text{BERT}_{\text{BASE}}$	Seq-Labeling	$O(n^2)$	91.9	50.8	80	84.6	51.1	80
Pointer-based FG22 w/ $\text{BERT}_{\text{BASE}}$	Multitask†	$O(n^2)$	–	–	–	89.8	**71.0**	–
Corro (2020) w/ BERT	Chart	$O(n^3)$	94.8	68.9	–	**90.0**	62.1	–
Ruprecht and Mörbitz (2021) w/ BERT	Chart	–	93.3	80.5	57	88.3	69.0	60
FG21 w/ XLNet (en) or $\text{BERT}_{\text{BASE}}$ (de)	Reorder-Chart	$O(n^3)$	95.1	74.1	179	88.5	63.0	238
FG21 w/ XLNet (en) or $\text{BERT}_{\text{BASE}}$ (de)	Reorder-Trans	$O(n^2)$	**95.5**	73.4	133	88.5	62.7	157
DB w/ XLNet (en) or $\text{BERT}_{\text{BASE}}$ (de)	Combinator	$O(n^2)$	94.8	76.6	**275**	89.5	69.7	**424**
DM w/ XLNet (en) or $\text{BERT}_{\text{BASE}}$ (de)	Combinator	$O(n^3)$	95.0	**83.0**	**375**	89.6	70.9	**535**

Table 3: Overall performance of recent discontinuous parsers. Speeds in sentences per second were obtained in tests conducted on incomparable hardware and software platforms. Ours and Vilares and Gómez-Rodríguez (2020, VG20) were conducted on a GeForce GTX 1080 Ti with a PyTorch implementation, and that of Fernández-González and Gómez-Rodríguez (2021, FG21) was conducted on a GeForce RTX 3090. Fernández-González and Gómez-Rodríguez (2022, FG22) involved lexical dependency information†.

DPTB (test)		**F1**	**D.F1**
PWE	**DB**	91.97±0.05	75.62±0.82
	DM	92.06±0.10	78.14±0.69
PLM	**DB**	94.84±0.24	76.62±2.07
	DM	95.04±0.06	83.04±0.79
TIGER (test)		**F1**	**D.F1**
PWE	**DB**	84.88±0.08	60.08±0.37
	DM	85.11±0.13	62.02±0.71
PLM	**DB**	89.48±0.16	69.68±0.55
	DM	89.61±0.09	70.93±0.63

Table 4: Means and standard deviations of five runs on test sets with four significant digits. **DM** outperforms **DB**. Development sets reflect similar variability.

Model		**DPTB (dev)**		**TIGER (dev)**	
(Stage)		**F1**	**D.F1**	**F1**	**D.F1**
DB	$(0,0,0)$	90.93	63.28	87.73	56.49
	$(0,1,1)$	91.61	69.84	88.70	61.15
	$(1,0,1)$	91.62	**74.25**	87.93	59.85
	$(1,1,0)$	91.48	70.97	89.05	63.32
(S1)	‡ (1, 1, 1)	**91.72**	66.82	**89.28**	**63.49**
(S2)	↪ *optuna*	92.25	76.60	89.59	66.03
DM	$(0,0,0)$	91.62	79.37	88.30	62.41
	$(0,1,1)$	91.44	78.70	88.61	65.10
	$(1,0,1)$	91.74	79.02	89.64	67.40
	$(1,1,0)$	91.84	77.37	**89.78**	67.78
(S1)	‡ (1,1,1)	**92.16**	**80.29**	89.77	**68.20**
(S2)	↪ *optuna*	92.37	82.76	89.84	68.45

Table 5: Ablation in two-stage training on development F1 scores. Triplets in $\{0,1\}$ indicate turning on and off ($\rho_\varnothing$, $\rho^{\textbf{DB}} \sim \text{Beta}(1,1)$, *shuffle*) for **DB** and ($\rho_\varnothing$, β_c, β_x) for **DM**. Variants of ''‡'' are **S1**—the start of **S2**.

conducted on different platforms, our parsers lead by a significant margin. In terms of overall F1 score, our parsers outperform some chart parsers (Stanojevic and Steedman, 2020; Ruprecht and Mörbitz, 2021) and slightly underperform the overall best outline, as characterized in **boldface**.

4.2 Ablation Study

We ablate the PWE models in two-stage training, as shown in Table 5. We only show one representative run with ablation because of the similar low variability on development sets. **DB** has two data augmentation items $\rho_\varnothing$ and $\rho^{\textbf{DB}}$ as well as one model item *ply shuffle*. On $\rho_\varnothing$ refers to $\rho_\varnothing = 0.25$ and off $\rho_\varnothing = 0$. Off $\text{Beta}(1,1)$ refers to a static

Model	Dev set	$\rho_\varnothing = 0$	0.1	‡0.25	0.5
DB	**DPTB**	91.61	91.79	91.72	**91.95**
	TIGER	88.70	89.04	**89.28**	89.25
DM	**DPTB**	91.44	91.80	**92.16**	89.86
	TIGER	88.61	89.45	**89.77**	88.61

Table 6: **DM** is sensitive to $\rho_\varnothing$ with dev F1 scores. All variants are based on $(\rho_\varnothing, 1, 1)$ in Table 5 with specific $\rho_\varnothing$ as the variable for **DB** and **DM**.

Test set	DM Medoid $\rho^{\mathbf{DM}}$	F1	D.F1
DPTB	*uhead*	**95.05**	**83.58**
	leftmost	95.00	81.64
	rightmost	95.03	82.47
	random (min)	95.01	82.18
	random (max)	95.04	83.17
TIGER	*uhead*	**89.62**	**71.61**
	leftmost	89.56	71.43
	rightmost	89.56	70.92
	random (min)	89.55	71.26
	random (max)	89.61	71.52

Table 7: **DM** medoid factor $\rho^{\mathbf{DM}} = \textit{uhead}$ offers stable gains even without head information during training. We tested $\rho^{\mathbf{DM}} = \textit{random}$ five times.

$\rho^{\mathbf{DB}} = 0.5$ and $(0, 0, 0)$ shows the performance of bare **DB** models.

On the flip side, **DM**'s $(0, 0, 0)$ contains randomness because of $\rho^{\mathbf{DM}} = \textit{random}$. We do not intend to examine a static $\rho^{\mathbf{DM}}$ as **DB** yields negative results. Based on effective training tricks, the variants enter the BO process at **S2**. **DCCP** shows its sensitivity to $\rho_\varnothing$ in Table 6.

4.3 Inference with Unsupervised Headedness

Both **CM** and **DM** provide unsupervised headedness $\bar{\lambda}$. Chen et al. (2021) were unable to test the benefits of **CM**'s unsupervised headedness because it is a final product that cannot affect parsing. However, **DM**'s medoid affects parsing performance. On PLM **DM**, we select different $\rho^{\mathbf{DM}}$ categories, which affect the location of all discontinuous constituent, and examine their generalization on test sets, as shown in Table 7. All models are trained with $\rho^{\mathbf{DM}} = \textit{random}$ but inference with $\rho^{\mathbf{DM}} = \textit{uhead}$ exerts positive gains on accuracy.

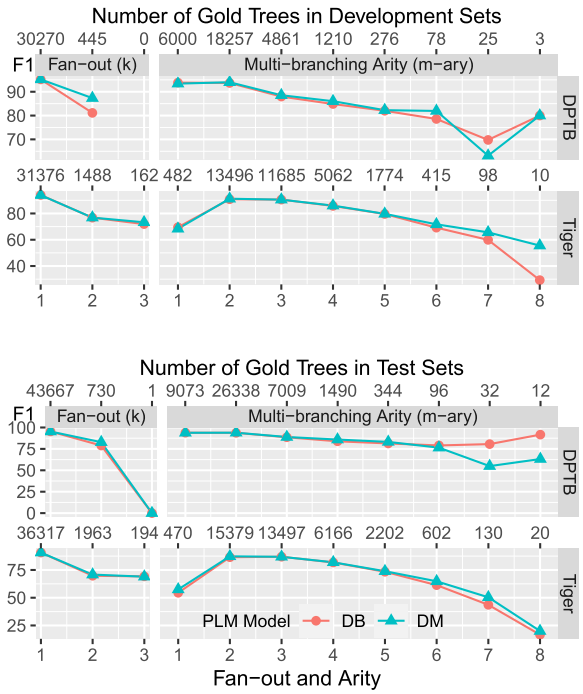


Figure 8: **DB** and **DM**'s discontinuity and multi-branching performance. Because the TIGER Treebank is richer in discontinuity, **DM** exhibits higher F1 scores.

5 Discussion

Properties of DCCP Models. Table 3 exhibits high speeds and near state-of-the-art accuracies of **DCCP** compared to recent works. **DCCP** inherits many properties from **NCCP**. **CB** and **CM** are special cases of **DB** and **DM** without *swap* and *discontinuous* actions. All models contain compact components without grammar restriction. Each model has no more than 4.7M parameters apart from PWE or PLM, as listed in Table 2.

The variability of all models is low, except for the discontinuous F1 score of PLM **DB** on DPTB, as shown in Table 4. The main cause may not be the random initialization but the different training processes of PWE and PLM models—PLM models use the configuration of PWE models at **S2**. Those degraded PLM models adopted low $\rho_\varnothing$ configurations (e.g., $\rho_\varnothing = 0.078$). Because DPTB has less discontinuity and overall F1 scores are used for model evaluation, high variability in discontinuous F1 scores becomes more common without several BO trials; this phenomenon is also reflected in Table 5. **DB** lacks explicit discontinuity, and the selection of hyperparameters seems to be necessary on DPTB.

Figure 8 presents the F1 scores for discontinuity and multi-branching. We select PLM models

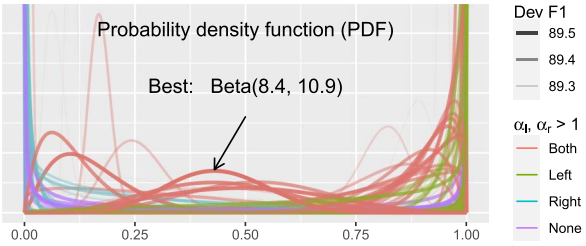


Figure 9: Beta distribution visualization for TIGER **DB** at **S2**. See their hyperparameters in Figure 10.

Parent (#)	**Head child by maximum weight**
NP (14.4K) from **CM**	DT (4.5K); NP (4.3K); NNP (1.6K); JJ (922); NN (751); NNS (616); etc. (1.6K; 12 of 50 types with ‘‘*’’)
NP(14.3K) from **DM**	NP (4.7K); DT (4.5K); NNP (1.6K); JJ (786); NN (715); NNS (565); etc. (1.4K; 15 of 49 types with ‘‘*’’)

Table 8: **CM** & **DM** unsupervised NP headedness from (D)PTB test sets. ‘‘*’’ denotes minority NPs having DTs as non-head children (i.e., DTs are strong NP heads).

whose overall F1 scores are close (i.e., most F1 differences are less than 0.1 and **DB**’s performance is high). **DM** exhibits persistent advantages over **DB** when these properties are frequent. We further determined that **CM** has the same gains over **CB** starting identically from 4-ary nodes with minor score differences on (D)PTB under the same $\rho_\varnothing = 0$ condition, as shown in Table 9. The result supports the argument of Xin et al. (2021), which asserts that m-ary constituency parsing without binarization preserves some natural advantages, e.g., predicate-argument structure. Specifically, $\rho_\varnothing > 0$ shifts **DM**’s multi-branching advantage to frequent low-arity trees, favoring the overall scores on DPTB, while it enhances both discontinuity and multi-branching advantages on TIGER, as shown in Table 10, in agreement with Table 6.

The training process of **CB** with CNF binarization has a slight impact on parsing accuracy. Chen et al. (2021) obtained the best **CB** with Bernoulli distribution $P(\rho^{\mathbf{DB}} = 0) = 0.85$ (i.e., $P(\rho^{\mathbf{DB}} = 1) = 0.15$ or L85R15 in their format ‘‘L%R%’’) on PTB. They argued that such binarization brings orientation balance.

Similarly, **DB**’s **S2** exhibits a slightly leftward exploration with the beta distribution, as shown in Figure 9. DPTB has a similar situation. Yet, the optimized distributions are relatively uniform and symmetric, which qualifies our uniform randomness with $\rho^{\mathbf{DB}}$ at **S1** and indicates a desired property for future language-agnostic practice.

For unsupervised headedness, Chen et al. (2021) reported that **CM** absolutely picks determiners (DT) as heads for major NP. **DM** continues and further grammatically picks more noun phrases (NP) as NP heads, as shown in Table 8.

Linguistic Properties by DCCP. From Figure 8, we learned that TIGER is more challenging in discontinuity. Incorrect discontinuity prediction seems to cascade to multi-branching prediction, resulting in degradation of both properties. Meanwhile, DPTB is largely transformed from PTB by typed traces and automatic rules, where the multi-branching accuracy stays more stable.

As seen in Table 7, $\rho^{\mathbf{DM}} = \textit{rightmost}, \textit{leftmost}$ yielded the second best F1 and D.F1 scores on DPTB and TIGER, respectively. The reversed setting yielded poor results, even if not the worst. The observation implies that many English heads locate rightward (right-branching) and that German heads tend to locate leftward (verb-second word order, V2). German has abundant separable verbs with their prefixes at the right-hand side of the clauses.

Error Rates of DCCP. Greedy parsers allow ill-formed outputs without a single root, especially in case of single-model inference. Our models yielded a few invalid parses, as demonstrated in Table 11. **DM** models produce more errors. However, unsuccessful decomposition of biaffine attention matrices might not be the direct cause, as also shown in Figure 11. $\rho_\varnothing > 0$ variants cleared the matrices that could not be decomposed with any θ (FAIL). Similar to **CM**, this genre suffers from more failures. Specifically, the ply size cannot be reduced to one during the iteration. Greedy parsers must suffer the defect because of their simplicity. However, invalid parses can contribute positive F1 scores and global parsers can yield inaccurate parses.

We applied methods such as Boolean matrix factorization and singular value decomposition. However, they did not provide any improvement and significantly slowed down the speed. This is

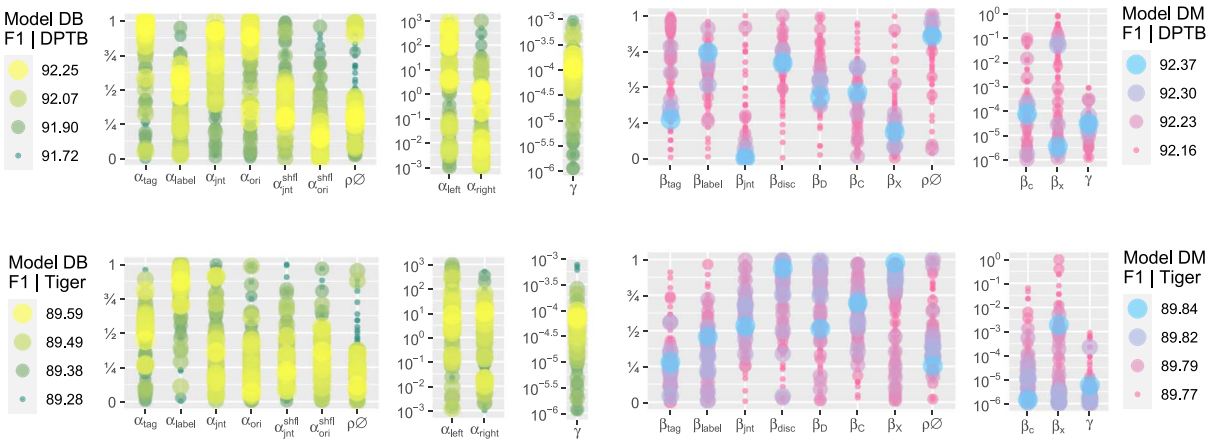


Figure 10: The BO process starts with **S1** dev F1 scores (i.e., a small dot at each legend bottom) and ends with a range of scores in **S2**. While the models are not sensitive to hyperparameters (e.g., all gains are less than 0.54), their preferences are different on respective corpora. On TIGER, $\alpha_{ori} < \alpha_{ori}^{shfl}$ and high (β_C, β_X) are preferable.

Prop.	**Gold**	**PWE**		$\rho_\varnothing = 0$		$\rho_\varnothing > 0$		**PLM**		$\rho_\varnothing = 0$		$\rho_\varnothing > 0$	
M**-ary**	**Tree**	**CB**	**CM**	**DB**	**DM**	**DB**	**DM**	**CB**	**CM**	**DB**	**DM**	**DB**	**DM**
1	9,073	**92.25**	92.02	**91.43**	91.35	**91.68**	91.53	93.80	**94.33**	93.36	**93.41**	**93.91**	93.82
2	26,338	**90.41**	89.94	**89.87**	89.68	89.95	**90.02**	**94.41**	94.33	93.47	**93.65**	93.77	**93.92**
3	7,009	**84.17**	83.56	**83.50**	83.34	83.60	**83.79**	**90.27**	89.81	88.31	**88.60**	88.57	**88.87**
4	1,490	77.87	**78.95**	78.19	**79.82**	78.50	**78.86**	**87.42**	86.51	83.98	**86.50**	83.88	**85.90**
5	344	74.19	**77.29**	76.14	**78.46**	74.97	**78.87**	81.42	**84.06**	78.61	**85.15**	81.38	**83.17**
6	96	70.05	**78.35**	72.90	**80.63**	**77.39**	76.29	78.64	**80.00**	79.23	**83.50**	**79.02**	76.44
7	32	64.71	**86.15**	73.53	**87.10**	**76.47**	71.43	**76.47**	70.18	75.76	**77.42**	**80.60**	54.90
8	12	64.00	**72.73**	81.82	**85.71**	75.00	**80.00**	78.26	**86.96**	78.57	**83.33**	**91.67**	63.16
9	3	**100.00**	75.00	**100.00**	75.00	**100.00**	50.00	**100.00**	75.00	**100.00**	85.71	85.71	**100.00**
$k > 1$	731	–	–	73.95	**77.60**	75.68	**78.94**	–	–	78.62	**83.04**	78.65	**82.71**
All	44,397	**92.54**	92.08	91.99	**92.02**	92.00	**92.00**	**95.71**	95.44	94.70	**94.79**	95.08	**95.09**

Table 9: Multi-branching and discontinuous F1 scores of **NCCP** and **DCCP** on (D)PTB test sets. We grouped $k > 1$ because only one tree has fan-out $k = 2$ in the test set. The scores of **CB** and **CM** are from Chen et al. (2021).

Prop.	**Gold**	**PWE** $\rho_\varnothing = 0$		$\rho_\varnothing > 0$		**PLM** $\rho_\varnothing = 0$		$\rho_\varnothing > 0$	
M**-ary**	**Tree**	**DB**	**DM**	**DB**	**DM**	**DB**	**DM**	**DB**	**DM**
1	470	45.37	**49.14**	51.64	**55.32**	55.16	**56.84**	54.45	**57.48**
2	15,379	81.83	**82.57**	82.25	**83.36**	85.91	**86.34**	86.50	**87.31**
3	13,497	**80.58**	80.41	80.95	**81.05**	**85.96**	85.35	**86.93**	86.89
4	6,166	**73.43**	73.34	74.04	**74.36**	**80.71**	79.93	81.76	**81.92**
5	2,202	63.66	**64.14**	64.27	**65.15**	71.09	**72.40**	73.37	**73.88**
6	602	50.72	**52.85**	51.62	**55.13**	59.30	**63.61**	61.32	**64.79**
7	130	36.25	**43.38**	40.53	**44.91**	45.51	**55.02**	43.59	**50.37**
8	20	11.24	**24.39**	16.44	**19.51**	**24.32**	24.24	16.67	**20.00**
9	6	12.90	**66.60**	21.43	**28.57**	16.67	**33.33**	12.90	**54.55**
$k = 1$	36,317	**85.95**	85.92	**86.41**	86.39	**89.85**	89.75	**90.69**	90.66
$k = 2$	1,963	**59.88**	59.64	59.95	**62.05**	**71.15**	67.53	69.78	**70.85**
$k = 3$	194	57.46	**59.83**	58.89	**60.61**	**68.23**	63.91	**69.21**	68.95
All	38,474	**84.56**	84.50	84.99	**85.08**	**88.82**	88.57	89.55	**89.58**

Table 10: Multi-branching and discontinuous test F1 scores of **DCCP** on TIGER. Fan-out is detailed in k.

Test set	DPTB	TIGER
Total trees	2,416	4,998
DB's ill-formed parses	1	2
DM's ill-formed parses	15	47
Biaffine attention matrices	594	7,278
$\theta = 0.5$ solutions	587	7,114
Average of tries if $\theta \neq 0.5$	12.4	42.9
FAIL + identity matrices	0+4	0+81

Table 11: Errors in **DCCP** PLM models with $\rho_{\varnothing} > 0$. A FAIL causes a matrix of ones, whereas a θ close to one yields an identity matrix—an expensive null action.

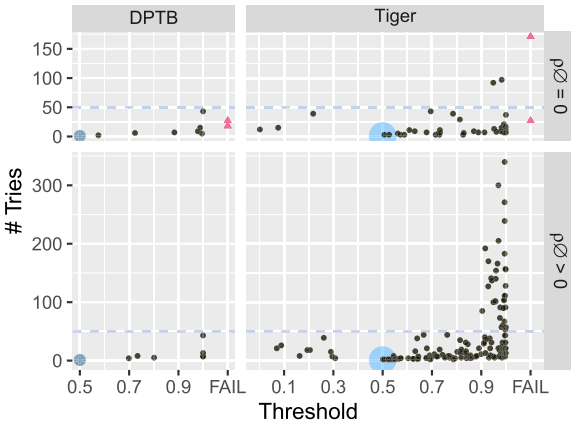


Figure 11: The numbers of tries to decompose biaffine attention matrices on test sets. "▲" marks FAILs.

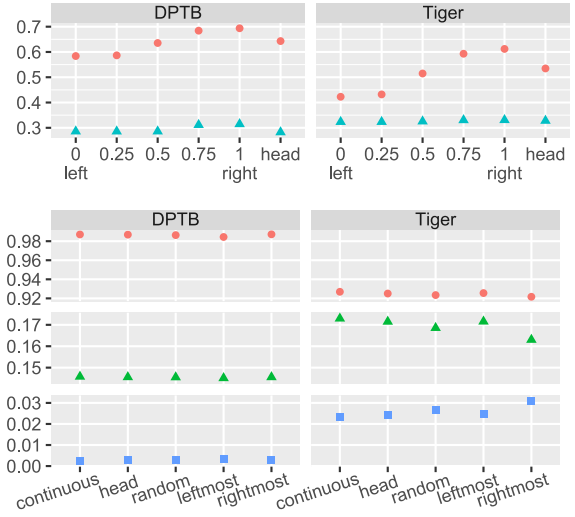


Figure 12: Signal polarity in corpora DPTB (sections 2–24) and TIGER. Top: **DB** signal polarity to $\rho^{\mathbf{DB}}$ with *orientation right* "•" and *joint* "▲" Bottom: **DM** signal polarity to stratifying *medoid* factor $\rho^{\mathbf{DM}}$ with *affinity* "•," *joint* "▲" and *discontinuity* "■." *Continuous* and *head* are referential only, looking for the least *discontinuity* and leveraging head information. (All $\rho_{\varnothing} = 0$.)

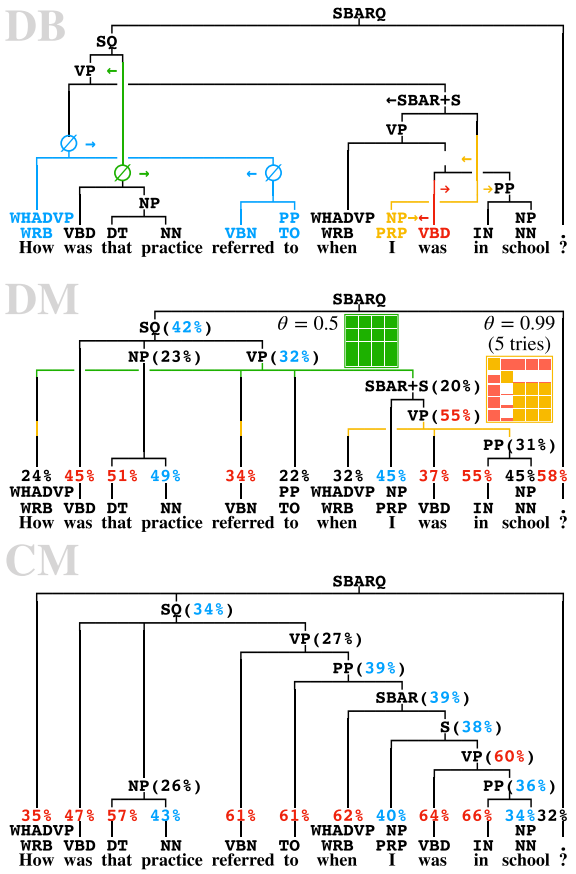


Figure 13: An exact matched DPTB sample from PLM **DB** and **DM** models versus **CM** on PTB. The parse contains complex nested clauses that **CM** must fail to capture, and it becomes ungrammatical in the continuous scenario. **DB**'s outputs include orientations depicted as arrows and their traveling traces colored for groups. Meanwhile, **DM** produces two biaffine attention matrices, one of which has a highly biased but correct threshold $\theta = 0.99$. Bar heights indicate values in matrices and their colors indicate the relationship to θ.

because $\theta \neq 0.5$ cases are few. Our sequential tries to decompose might be naïve, but they are effective. In Figure 11, the number of tries does not significantly increase under $\theta < 0.9$ within 50 tries. For $\theta \geq 0.9$, although some tries are expensive, we will see that they are worthy in the next section. The imbalanced signals from both datasets account for the bias of θ—more than 92% of the biaffine attention signals are ones, as shown in Figure 12. All affinity biases (i.e., $b^{aff} \in [-1.60, -0.84]$) are significant negative numbers for counteraction.

Weakness. The design of *affinity* as biaffine attention is an initial but coarse attempt, which

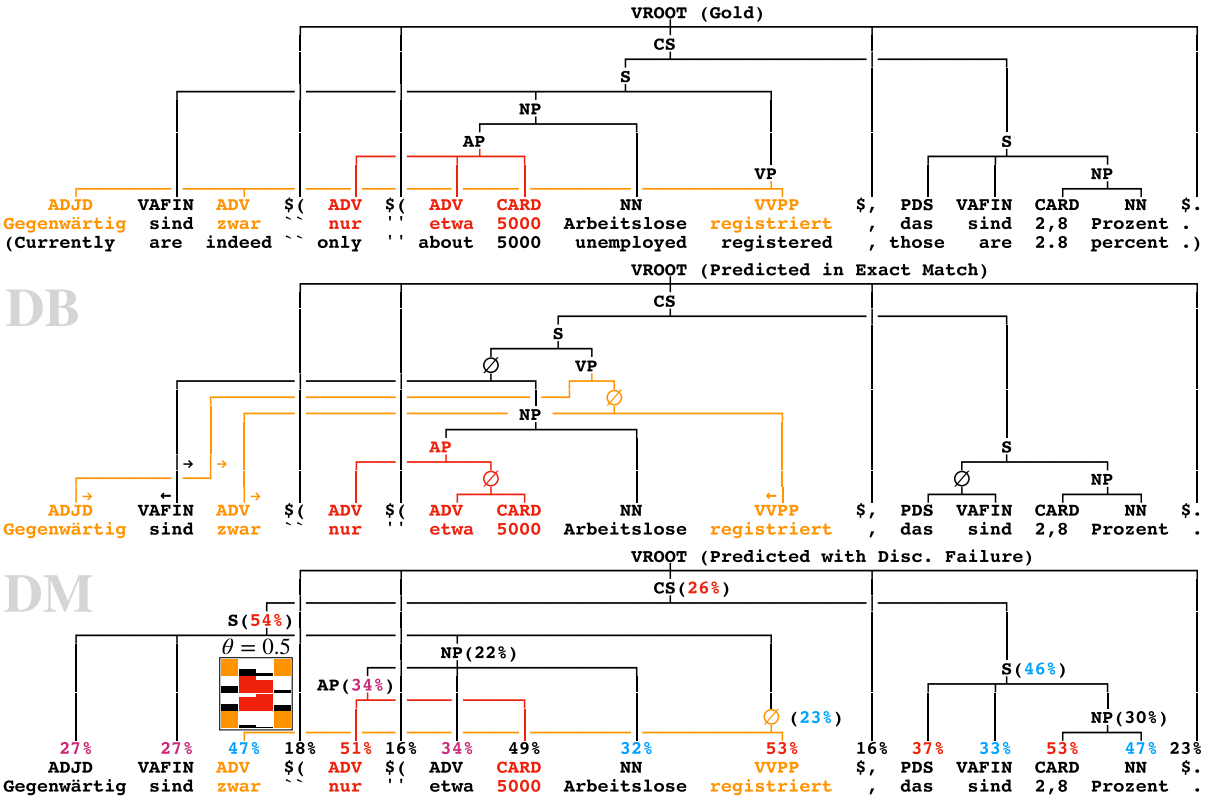


Figure 14: A TIGER parse. **DB** natively with ∅-subtrees achieved the exact match but **DM** erred with $\rho_{\varnothing} = 0$.

brings imbalance. If one encoded *dependency within constituent* to biaffine attention instead, both signal balance and multi-grammar parsing might be better addressed. However, based on the context of constituency parsing, we leave this topic for future study.

6 Sample Analysis

Continuous vs. Discontinuous Parsing. Figure 13 highlights the value of discontinuous parsing by demonstrating the respective **CM** parse. Conspicuously, the branching tendency of the continuous parse is to the right, while it is not obvious for the discontinuous parsing. Meanwhile, we observed instances of similar unsupervised headedness weights. This sample is not trivial, which challenges our PLM **DCCP** models.

Parsing Process of DCCP. In Figure 13, **DB** shows sinuous travel traces of ‘‘I,’’ ‘‘was,’’ and larger ∅-subtree nodes that involve the turning of orientations. The varying context leads them to achieve complex movement. **DB** also created some grammatical substructures for ‘‘How,’’ ‘‘referred,’’ ‘‘to,’’ ‘‘was,’’ ‘‘in,’’ and ‘‘school.’’

Meanwhile, the **DM** parse is more dramatic. The formation of the lower discontinuous VP involves five nodes, two of which are irrelevant words ‘‘How’’ and ‘‘referred’’ triggered by incorrect *discontinuity* signals. They are discontinuous but for the higher VP. The two nodes create a noisy biaffine attention matrix because their grammatical roles are compatible with the lower VP. Trained for extra robustness, the matrix decomposition with five tries found the right θ to identify the correct VP members excluding ‘‘How’’ and ‘‘referred.’’ The interply loss and the decoding process gave this parse a chance for perfection.

In Figure 14 for German, **DB** achieved a long distant constituent in a more subtle way. The word ‘‘zwar’’ joins ‘‘registriert’’ as an ∅-subtree when the formation of intermediate NP shortens their distance and prevents a travel through. ‘‘Gegenwärtig’’ follows and forms a VP. However, **DM** failed.

The $\rho_{\varnothing} > 0$ matters. The above failure explains why **DM** is inferior to **DB** with $\rho_{\varnothing} = 0$. **DB**’s orientation system allows some free travel before joining with correct mates. The constituent formation through steps of accumulation creates

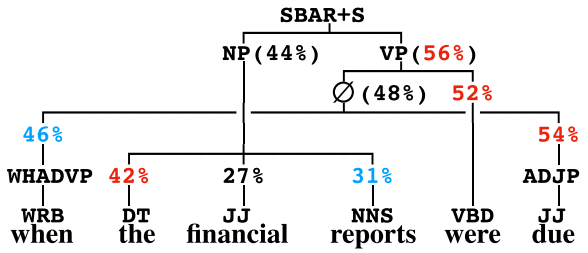


Figure 15: A semantic ∅-subtree by **DM** with $\rho_{\varnothing} > 0$. Copula ‘‘were’’ has less *affinity* than ‘‘when’’ and ‘‘due.’’

a more stable context. However, **DM**’s group action happens all at once. An incorrect composition might create a quite different context, which leads to unseen reaction chains. The strange unsupervised headedness weights reflect the issue. On the other side, with $\rho_{\varnothing} > 0$, **DM** can also gradually build and discover some semantic substructures, as shown in Figure 15. In contrast, **DB** is not sensitive to $\rho_{\varnothing}$ because of its nature, in agreement with Table 6.

7 Conclusion

We proposed a pair of efficient and effective discontinuous combinatory constituency parsers and extended the neural combinator family of **NCCP**. A binary combinator **DB** is based on orientation extended into a joint-swap system. A multi-branching combinator **DM** leverages biaffine attention adapted for constituency. Our models (as in Table 2) achieved state-of-the-art discontinuous F1 scores with a significant advantage in speed.

In the future, we will aim to extend **DCCP** into a multilingual tool for directed acyclic graph (DAG) parsing with function tag prediction for predicate-argument structure.

Acknowledgments

We extend special thanks to our action editor, anonymous reviewers, and Prof. Yusuke Miyao for their invaluable comments and suggestions. This work was partly supported by TMU research fund for young scientists.

References

Takuya Akiba, Shotaro Sano, Toshihiko Yanase, Takeru Ohta, and Masanori Koyama. 2019. Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2623–2631. https://doi.org/10.1145/3292500.3330701

Sabine Brants, Stefanie Dipper, Peter Eisenberg, Silvia Hansen-Schirra, Esther König, Wolfgang Lezius, Christian Rohrer, George Smith, and Hans Uszkoreit. 2004. TIGER: Linguistic interpretation of a German corpus. *Research on Language and Computation*, 2(4):597–620. https://doi.org/10.1007/s11168-004-7431-3

Zhousi Chen, Longtu Zhang, Aizhan Imankulova, and Mamoru Komachi. 2021. Neural combinatory constituency parsing. In *Findings of the Association for Computational Linguistics: ACL/IJCNLP*, pages 2199–2213. https://doi.org/10.18653/v1/2021.findings-acl.194

Maximin Coavoux and Shay B. Cohen. 2019. Discontinuous constituency parsing with a stack-free transition system and a dynamic oracle. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 204–217. https://doi.org/10.18653/v1/n19-1018

Maximin Coavoux and Benoît Crabbé. 2017. Incremental discontinuous phrase structure parsing with the GAP transition. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics*, pages 1259–1270. https://doi.org/10.18653/v1/e17-1118

Maximin Coavoux, Benoît Crabbé, and Shay B. Cohen. 2019. Unlexicalized transition-based discontinuous constituency parsing. *Transactions of the Association for Computational Linguistics*, 7:73–89. https://doi.org/10.1162/tacl_a_00255

Ronan Collobert. 2011. Deep learning for efficient discriminative parsing. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, volume 15 of *JMLR Proceedings*, pages 224–232. http://proceedings.mlr.press/v15/collobert11a/collobert11a.pdf

Caio Corro. 2020. Span-based discontinuous constituency parsing: A family of exact chart-based algorithms with time complexities from O(n^6) down to O(n^3). In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*, pages 2753–2764. https://doi.org/10.18653/v1/2020.emnlp-main.219

Timothy Dozat and Christopher D. Manning. 2017. Deep biaffine attention for neural dependency parsing. In *5th International Conference on Learning Representations*.

Kilian Evang and Laura Kallmeyer. 2011. PLCFRS parsing of english discontinuous constituents. In *Proceedings of the 12th International Conference on Parsing Technologies*, pages 104–116.

Daniel Fernández-González and Carlos Gómez-Rodríguez. 2020a. Discontinuous constituent parsing with pointer networks. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence*, pages 7724–7731. https://doi.org/10.1609/aaai.v34i05.6275

Daniel Fernández-González and Carlos Gómez-Rodríguez. 2020b. Multitask pointer network for multi-representational parsing. *CoRR*, abs/2009.09730.

Daniel Fernández-González and Carlos Gómez-Rodríguez. 2021. Reducing discontinuous to continuous parsing with pointer network reordering. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 10570–10578. https://doi.org/10.18653/v1/2021.emnlp-main.825

Daniel Fernández-González and Carlos Gómez-Rodríguez. 2022. Multitask pointer network for multi-representational parsing. *Knowledge-Based Systems*, 236:107760. https://doi.org/10.1016/j.knosys.2021.107760

Mark Johnson. 1985. Parsing with discontinuous constituents. In *Proceedings of the 23rd Annual Meeting of the Association for Computational Linguistics*, pages 127–132. https://doi.org/10.3115/981210.981225

Dan Jurafsky and James H. Martin. 2009. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition, 2nd Edition*. Prentice Hall series in artificial intelligence, Prentice Hall, Pearson Education International.

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. https://doi.org/10.18653/v1/P18-1249

Nikita Kitaev and Dan Klein. 2020. Tetra-tagging: Word-synchronous parsing with linear-time inference. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6255–6261. https://doi.org/10.18653/v1/2020.acl-main.557

Nikita Kitaev, Thomas Lu, and Dan Klein. 2022. Learned incremental representations for parsing. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics*, pages 3086–3095. https://doi.org/10.18653/v1/2022.acl-long.220

Wang Ling, Chris Dyer, Alan W. Black, and Isabel Trancoso. 2015. Two/too simple adaptations of word2vec for syntax problems. In *The 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1299–1304. https://doi.org/10.3115/v1/n15-1142

Wolfgang Maier. 2015. Discontinuous incremental shift-reduce parsing. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing*, pages 1202–1212. https://doi.org/10.3115/v1/p15-1116

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of English: The Penn Tree bank. *Computational Linguistics*, 19(2):313–330. https://doi.org/10.21236/ADA273556

Joakim Nivre, Marco Kuhlmann, and Johan Hall. 2009. An improved oracle for dependency

parsing with online reordering. In *Proceedings of the 11th International Workshop on Parsing Technologies*, pages 73–76. https://doi.org/10.3115/1697236.1697250

Adwait Ratnaparkhi. 1997. A linear observed time statistical parser based on maximum entropy models. In *Second Conference on Empirical Methods in Natural Language Processing*. https://aclanthology.org/W97-0301/

Thomas Ruprecht and Richard Mörbitz. 2021. Supertagging-based parsing with linear context-free rewriting systems. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2923–2935. https://doi.org/10.18653/v1/2021.naacl-main.232

Yikang Shen, Zhouhan Lin, Athul Paul Jacob, Alessandro Sordoni, Aaron C. Courville, and Yoshua Bengio. 2018. Straight to the tree: Constituency parsing with neural syntactic distance. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*, pages 1171–1180. https://doi.org/10.18653/v1/P18-1108

Milos Stanojevic and Mark Steedman. 2020. Span-based LCFRS-2 parsing. In *Proceedings of the 16th International Conference on Parsing Technologies and the IWPT 2020 Shared Task on Parsing into Enhanced Universal Dependencies*, pages 111–121. https://doi.org/10.18653/v1/2020.iwpt-1.12

Yannick Versley. 2014. Incorporating semi-supervised features into discontinuous easy-first constituent parsing. *CoRR*, abs/1409.3813.

David Vilares and Carlos Gómez-Rodríguez. 2020. Discontinuous constituent parsing as sequence labeling. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*, pages 2771–2785. https://doi.org/10.18653/v1/2020.emnlp-main.221

Xin Xin, Jinlong Li, and Zeqi Tan. 2021. N-ary constituent tree parsing with recursive semi-Markov model. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing*, pages 2631–2642. https://doi.org/10.18653/v1/2021.acl-long.205

Ryo Yoshida, Hiroshi Noji, and Yohei Oseki. 2021. Modeling human sentence processing with left-corner recurrent neural network grammars. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 2964–2973. https://doi.org/10.18653/v1/2021.emnlp-main.235

Junru Zhou and Hai Zhao. 2019. Head-driven phrase structure grammar parsing on Penn Treebank. In *Proceedings of the 57th Conference of the Association for Computational Linguistics*, pages 2396–2408. https://doi.org/10.18653/v1/P19-1230

Arnold M. Zwicky. 1985. Heads. *Journal of Linguistics*, 21(1):1–29. https://doi.org/10.1017/S0022226700010008