Transition-based Bubble Parsing: Improvements on Coordination Structure Prediction

Tianze Shi
Cornell University
tianze@cs.cornell.edu

Lillian Lee
Cornell University
llee@cs.cornell.edu

Abstract

We propose a transition-based bubble parser to perform coordination structure identification and dependency-based syntactic analysis simultaneously. Bubble representations were proposed in the formal linguistics literature decades ago; they enhance dependency trees by encoding coordination boundaries and internal relationships within coordination structures explicitly. In this paper, we introduce a transition system and neural models for parsing these bubble-enhanced structures. Experimental results on the English Penn Treebank and the English GENIA corpus show that our parsers beat previous state-of-the-art approaches on the task of coordination structure prediction, especially for the subset of sentences with complex coordination structures.[1]

1 Introduction

Coordination structures are prevalent in treebank data (Ficler and Goldberg, 2016a), especially in long sentences (Kurohashi and Nagao, 1994), and they are among the most challenging constructions for NLP models. Difficulties in correctly identifying coordination structures have consistently contributed to a significant portion of errors in state-of-the-art parsers (Collins, 2003; Goldberg and Elhadad, 2010; Ficler and Goldberg, 2017). These errors can further propagate to downstream NLP modules and applications, and limit their performance and utility. For example, Saha et al. (2017) report that missing conjuncts account for two-thirds of the errors in recall made by their open information extraction system.

Coordination constructions are particularly challenging for the widely-adopted dependency-based paradigm of syntactic analysis, since the asymmetric definition of head-modifier dependency relations is not directly compatible with the symmetric nature of the relations among the participating conjuncts and coordinators.[2] Existing treebanks usually resort to introducing special relations to represent coordination structures. But, there remain theoretical and empirical challenges regarding how to most effectively encode information like modifier sharing relations while still permitting accurate statistical syntactic analysis.

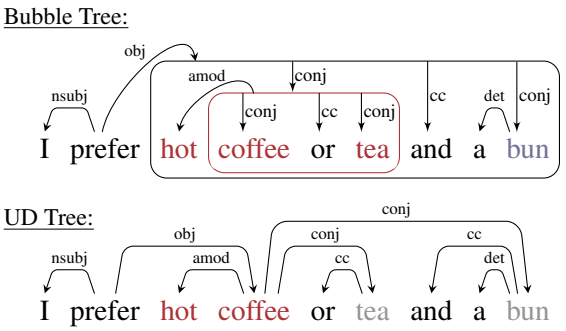

Figure 1: Bubble tree and (basic) UD tree for the same example sentence. (For clarity, we omit punctuation and single-word bubble boundaries.) Bubbles explicitly encode the scope of the shared modifier *hot* with respect to the nested coordination, whereas the UD tree gives both *tea* and *bun* identical relationships to *hot*.

In this paper, we explore Kahane's (1997) alternative solution: extend the dependency-tree representation by introducing *bubble structures* to explicitly encode coordination boundaries. The co-heads within a bubble enjoy a symmetric relationship, as befits a model of conjunction. Further, bubble trees support representation of nested coordination, with the scope of shared modifiers identifiable by the attachment sites of bubble arcs. Figure 1 compares a bubble tree against a Universal Dependencies (UD; Nivre et al., 2016, 2020) tree for the same sentence.

Yet, despite theses advantages, implementation

[1] Code at github.com/tzshi/bubble-parser-acl21.

[2] Rambow (2010) comments on other divergences between syntactic representation and syntactic phenomena.

Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, pages 7167–7182
August 1–6, 2021.

of the formalism was not broadly pursued, for reasons unknown to us. Given its appealing and intuitive treatment of coordination phenomena, we revisit the bubble tree formalism, introducing and implementing a transition-based solution for parsing bubble trees. Our transition system, Bubble-Hybrid, extends the Arc-Hybrid transition system (Kuhlmann et al., 2011) with three bubble-specific transitions, each corresponding to opening, expanding, and closing bubbles. We show that our transition system is both sound and complete with respect to projective bubble trees (defined in § 2.2).

Experiments on the English Penn Treebank (PTB; Marcus et al., 1993) extended with coordination annotation (Ficler and Goldberg, 2016a) and the English GENIA treebank (Kim et al., 2003) demonstrate the effectiveness of our proposed transition-based bubble parsing on the task of coordination structure prediction. Our method achieves state-of-the-art performance on both datasets and improves accuracy on the subset of sentences exhibiting complex coordination structures.

2 Dependency Trees and Bubble Trees

2.1 Dependency-based Representations for Coordination Structures

A dependency tree encodes syntactic relations via directed bilexical dependency edges. These are natural for representing argument and adjunct modification, but Popel et al. (2013) point out that "dependency representation is at a loss when it comes to representing paratactic linguistic phenomena such as coordination, whose nature is symmetric (two or more conjuncts play the same role), as opposed to the head-modifier asymmetry of dependencies" (pg. 517).

If one nonetheless persists in using dependency relations to annotate all syntactic structures, as is common practice in most dependency treebanks (Hajič et al., 2001; Nivre et al., 2016, *inter alia*), then one must introduce special relations to represent coordination structures and promote one element from each coordinated phrase to become the "representational head". One choice is to specify one of the conjuncts as the "head" (Mel'čuk, 1988, 2003; Järvinen and Tapanainen, 1998; Lombardo and Lesmo, 1998) (e.g., in Figure 1, the visually asymmetric "conj" relation between "coffee" and "tea" is overloaded to admit a symmetric relationship), but it is then non-trivial to distinguish shared modifiers from private ones (e.g., in the UD tree at the bottom of Figure 1, it is difficult to tell that "hot" is private to "coffee" and "tea", which share it, but "hot" does not modify "bun"). Another choice is let one of the coordinators dominate the phrase (Hajič et al., 2001, 2020), but the coordinator does not directly capture the syntactic category of the coordinated phrase. Decisions on which of these dependency-based fixes is more workable are further complicated by the interaction between representation styles and their learnability in statistical parsing (Nilsson et al., 2006; Johansson and Nugues, 2007; Rehbein et al., 2017).

Enhanced UD A tactic used by many recent releases of UD treebanks is to introduce certain extra edges and non-lexical nodes (Schuster and Manning, 2016; Nivre et al., 2018; Bouma et al., 2020). While some of the theoretical issues still persist in this approach with respect to capturing the symmetric nature of relations between conjuncts, this solution better represents shared modifiers in coordinations, and so is a promising direction. In work concurrent with our own, Grünewald et al. (2021) manually correct the coordination structure annotations in an English treebank under the enhanced UD representation format. We leave it to future work to explore the feasibility of automatic conversion of coordination structure representations between enhanced UD trees and *bubble trees*, which we discuss next.

2.2 Bubble Trees

An alternative solution to the coordination-in-dependency-trees dilemma is to permit certain restricted phrase-inspired constructs for such structures. Indeed, Tesnière's (1959) seminal work on dependency grammar does not describe all syntactic relations in terms of dependencies, but rather reserves a primitive relation for connecting coordinated items. Hudson (1984) further extends this idea by introducing explicit markings of coordination boundaries.

In this paper, we revisit *bubble trees*, a representational device along the same vein introduced by Kahane (1997) for syntactic representation. (Kahane credits Gladkij (1968) with a formal study.) *Bubbles* are used to denote coordinated phrases; otherwise, asymmetric dependency relations are retained. Conjuncts immediately within the bubble may co-head the bubble, and the bubble itself may establish dependencies with its governor and modifiers. Figure 1 depicts an example bubble tree.

We now formally define bubble trees and their projective subset, which will become the focus of our transition-based parser in §3. The following formal descriptions are adapted from Kahane (1997), tailored to the presentation of our parser.

Formal Definition Given a dependency-relation label set L, we define a bubble tree for a length-n sentence $W = w_1, \ldots, w_n$ to be a quadruple $(V, \mathcal{B}, \phi, A)$, where $V = \{\texttt{RT}, w_1, \ldots, w_n\}$ is the ground set of nodes (RT is the dummy root), $\mathcal{B}$ is a set of bubbles, the function $\phi : \mathcal{B} \mapsto (2^V \backslash \{\varnothing\})$ gives the content of each bubble as a non-empty[3] subset of V, and $A \subset \mathcal{B} \times L \times \mathcal{B}$ defines a labeled directed tree over $\mathcal{B}$. Given labeled directed tree A, we say $\alpha_1 \rightarrow \alpha_2$ if and only if $(\alpha_1, l, \alpha_2) \in A$ for some l. We denote the reflexive transitive closure of relation $\rightarrow$ by $\overset{*}{\rightarrow}$.

Bubble tree $(V, \mathcal{B}, \phi, A)$ is *well-formed* if and only if it satisfies the following conditions:[4]

- No partial overlap: $\forall \alpha_1, \alpha_2 \in \mathcal{B}$, either $\phi(\alpha_1) \cap \phi(\alpha_2) = \varnothing$ or $\phi(\alpha_1) \subseteq \phi(\alpha_2)$ or $\phi(\alpha_2) \subseteq \phi(\alpha_1)$;
- Non-duplication: there exists no non-identical $\alpha_1, \alpha_2 \in \mathcal{B}$ such that $\phi(\alpha_1) = \phi(\alpha_2)$;
- Lexical coverage: for any singleton (i.e., one-element) set s in 2^V, $\exists \alpha \in \mathcal{B}$ such that $\phi(\alpha) = s$;
- Roothood: the root RT appears in exactly one bubble, a singleton that is the root of the tree defined by A.
- Containment: if $\exists \alpha_1, \alpha_2 \in \mathcal{B}$ such that $\phi(\alpha_2) \subset \phi(\alpha_1)$, then $\alpha_1 \overset{*}{\rightarrow} \alpha_2$.

Projectivity Our parser focuses on the subclass of *projective* well-formed bubble trees. Visually, a projective bubble tree only contains bubbles covering a consecutive sequence of words (such that we can draw boxes around the span of words to represent them) and can be drawn with all arcs arranged spatially above the sentence where no two arcs or bubble boundaries cross each other. The bubble tree in Figure 1 is projective.

Formally, we define the projection $\psi(\alpha) \in 2^V$ of a bubble $\alpha \in \mathcal{B}$ to be all nodes the bubble and its subtree cover, that is, $v \in \psi(\alpha)$ if and only if $\alpha \overset{*}{\rightarrow} \alpha'$ and $v \in \phi(\alpha')$ for some α'. Then, we can define a well-formed bubble tree to be *projective* if and only if it additionally satisfies the following:

- Continuous coverage: for any bubble $\alpha \in \mathcal{B}$, if $w_i, w_j \in \phi(\alpha)$ and $i < k < j$, then $w_k \in \phi(\alpha)$;
- Continuous projections: for any bubble $\alpha \in \mathcal{B}$, if $w_i, w_j \in \psi(\alpha)$ and $i < k < j$, then $w_k \in \psi(\alpha)$;
- Contained projections: for $\alpha_1, \alpha_2 \in \mathcal{B}$, if $\alpha_1 \overset{*}{\rightarrow} \alpha_2$, then either $\psi(\alpha_2) \subset \phi(\alpha_1)$ or $\psi(\alpha_2) \cap \phi(\alpha_1) = \varnothing$.

3 Our Transition System for Parsing Bubble Trees

Although, as we have seen, bubble trees have theoretical benefits in representing coordination structures that interface with an overall dependency-based analysis, there has been a lack of parser implementations capable of handling such representations. In this section, we fill this gap by introducing a transition system that can incrementally build projective bubble trees.

Transition-based approaches are popular in dependency parsing (Nivre, 2008; Kübler et al., 2009). We propose to extend the Arc-Hybrid transition system (Kuhlmann et al., 2011) with transitions specific to bubble structures.[5]

3.1 Bubble-Hybrid Transition System

A transition system consists of a data structure describing the intermediate parser states, called *configurations*; specifications of the *initial* and *terminal configurations*; and an inventory of *transitions* that advance the parser in configuration space towards reaching a terminal configuration.

Our transition system uses a similar configuration data structure to that of Arc-Hybrid, which consists of a stack, a buffer, and the partially-committed syntactic analysis. Initially, the stack only contains a singleton bubble corresponding to $\{\texttt{RT}\}$, and the buffer contains singleton bubbles, each representing a token in the sentence. Then, through taking transitions one at a time, the parser can incrementally move items from the buffer to the stack, or reduce items by attaching them to other bubbles or merging them into larger bubbles. Eventually, the parser should arrive at a terminal configuration where the stack contains the singleton bubble of $\{\texttt{RT}\}$ again, but the buffer is empty as all the tokens are now attached to or contained in other bubbles that are now descendants of the

[3] Our definition does not allow empty nodes; we leave it to future work to support them for gapping constructions.

[4] We do not use β for bubbles because we reserve the β symbol for our parser's buffer.

[5] Our strategy can be adapted to other transition systems as well; we focus on Arc-Hybrid here because of its comparatively small inventory of transitions, absence of spurious ambiguities (there is a one-to-one mapping between a gold tree and a valid transition sequence), and abundance of existing implementations (e.g., Kiperwasser and Goldberg, 2016).

Transition (Pre-conditions)	From Stack σ	From Buffer β	To Stack σ'	To Buffer β'
SHIFT ($\lvert\beta\rvert \geq 1$)	$\ldots$	$b_1 \ldots$	$\ldots b_1$	$\ldots$
LEFTARC$_{\text{lbl}}$ ($\lvert\sigma\rvert \geq 1; \lvert\beta\rvert \geq 1; s_1, b_1 \notin \mathcal{O}; \phi(s_1) \neq \{\texttt{RT}\}$)	$\ldots s_1$	$b_1 \ldots$	$\ldots$	$b_1 \ldots$ (lbl: s_1)
RIGHTARC$_{\text{lbl}}$ ($\lvert\sigma\rvert \geq 2; s_1, s_2 \notin \mathcal{O}$)	$\ldots s_2\ s_1$	$\ldots$	$\ldots s_2$ (lbl: s_1)	$\ldots$
BUBBLEOPEN$_{\text{lbl}}$ ($\lvert\sigma\rvert \geq 2; s_1, s_2 \notin \mathcal{O}; \phi(s_2) \neq \{\texttt{RT}\}$)	$\ldots s_2\ s_1$	$\ldots$	$\ldots$ (conj: s_2, lbl: s_1)	$\ldots$
BUBBLEATTACH$_{\text{lbl}}$ ($\lvert\sigma\rvert \geq 2; s_1 \notin \mathcal{O}; s_2 \in \mathcal{O}$)	$\ldots$ (conj: $s_{2a} \ldots$) s_1	$\ldots$	$\ldots$ (conj: $s_{2a} \ldots$, lbl: s_1)	$\ldots$
BUBBLECLOSE ($\lvert\sigma\rvert \geq 1; s_1 \in \mathcal{O}$)	$\ldots$ (conj: $s_{1a} \ldots$)	$\ldots$	$\ldots$	(conj: $s_{1a} \ldots$) $\ldots$

Table 1: Illustration of our Bubble-Hybrid transition system. We give the pre-conditions for each transition and visualizations of the affected stack and buffer items comparing the configurations before and after taking that transition. $\mathcal{O}$ denotes the set of currently open bubbles and RT is the dummy root symbol.

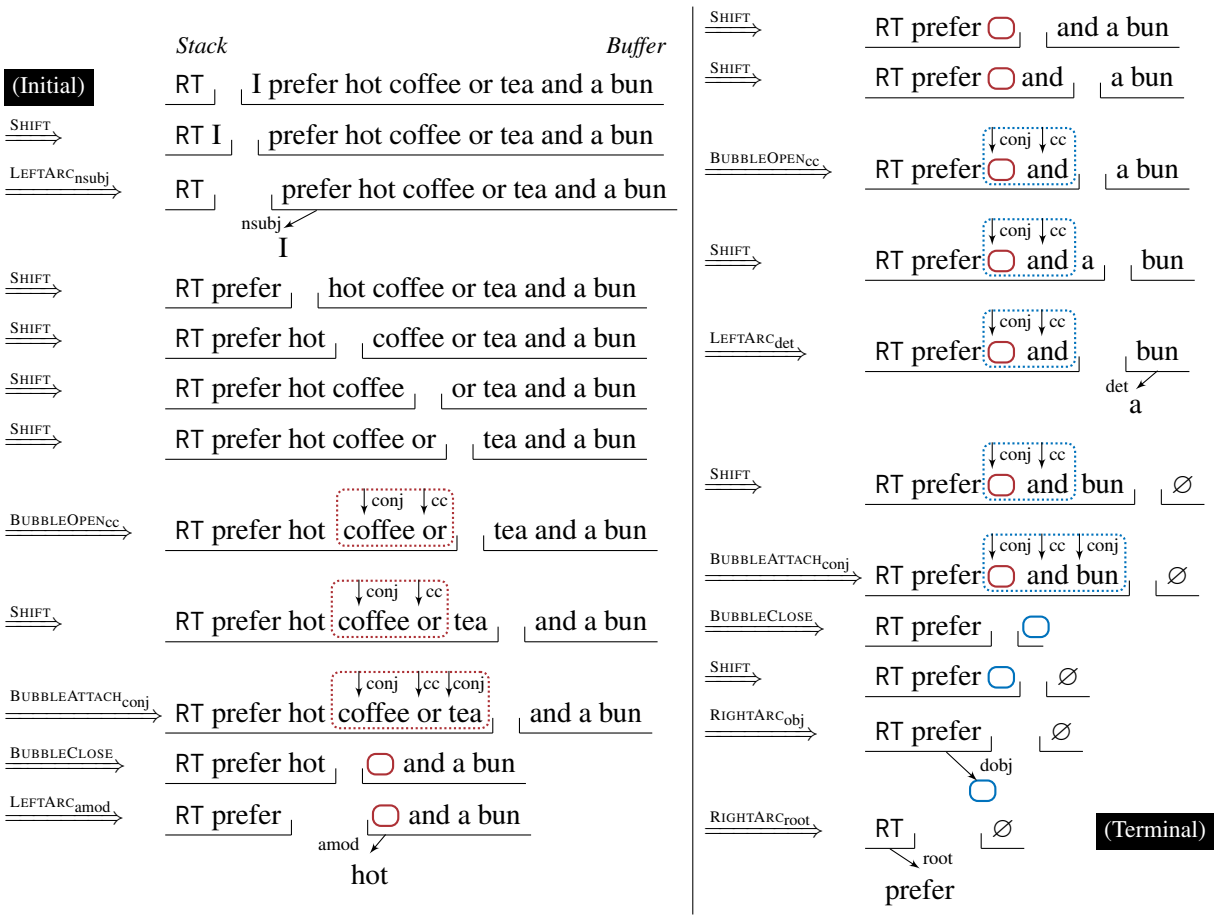

Figure 2: Step-by-step visualization of the stack and buffer during parsing of the example sentence in Figure 1. For steps following an attachment or BUBBLECLOSE transition, the detailed subtree or internal bubble structure is omitted for visual clarity. For the same reason, we omit drawing the boundaries around singleton bubbles.

{RT} singleton, and we can retrieve a completed bubble-tree parse.

Table 1 lists the available transitions in our Bubble-Hybrid system. The SHIFT, LEFTARC, and RIGHTARC transitions are as in the Arc-Hybrid system. We introduce three new transitions to handle coordination-related bubbles: BUBBLEOPEN puts the first two items on the stack into an open bubble, with the first item in the bubble, i.e., previously the second topmost item on the stack, labeled as the first conjunct of the resulting bubble; BUBBLEATTACH absorbs the topmost item on the stack into the open bubble that is at the second topmost position; and finally, BUBBLECLOSE closes the open bubble at the top of the stack and moves it to the buffer, which then allows it to take modifiers from its left through LEFTARC transitions. Figure 2 visualizes the stack and buffer throughout the process of parsing the example sentence in Figure 1. In particular, the last two steps in the left column of Figure 2 show the bubble corresponding to the phrase "coffee or tea" receiving its left modifier "hot" through a LEFTARC transition after it is put back on the buffer by a BUBBLECLOSE transition.

Formal Definition Our transition system is a quadruple (C, T, c^i, C_τ), where C is the set of configurations to be defined shortly, T is the set of transitions with each element being a partial function $t \in T : C \rightharpoonup C$, c^i maps a sentence to its intial configuration, and $C_\tau \subset C$ is a set of terminal configurations. Each configuration $c \in C$ is a septuple $(\sigma, \beta, V, \mathcal{B}, \phi, A, \mathcal{O})$, where V, $\mathcal{B}$, ϕ, and A define a partially-recognized bubble tree, σ and β are each an (ordered) list of items in $\mathcal{B}$, and $\mathcal{O} \subset \mathcal{B}$ is a set of open bubbles. For a sentence $W = w_1, \ldots, w_n$, we let $c^i(W) = (\sigma^0, \beta^0, V, \mathcal{B}^0, \phi^0, \{\}, \{\})$, where $V = \{\mathsf{RT}, w_1, \ldots, w_n\}$, $\mathcal{B}^0$ contains $n+1$ items, $\phi^0(\mathcal{B}^0_0) = \{\mathsf{RT}\}$, $\phi^0(\mathcal{B}^0_i) = \{w_i\}$ for i from 1 to n, $\sigma^0 = [\mathcal{B}^0_0]$, and $\beta^0 = [\mathcal{B}^0_1, \ldots, \mathcal{B}^0_n]$. We write $\sigma|s_1$ and $b_1|\beta$ to denote a stack and a buffer with their topmost items being s_1 and b_1 and the remainders being σ and β respectively. We also omit the constant V in describing c when the context is clear.

For the transitions T, we have:

- SHIFT$[(\sigma, b_1|\beta, \mathcal{B}, \phi, A, \mathcal{O})] =$ $(\sigma|b_1, \beta, \mathcal{B}, \phi, A, \mathcal{O})$;
- LEFTARC$_{\text{lbl}}[(\sigma|s_1, b_1|\beta, \mathcal{B}, \phi, A, \mathcal{O})] =$ $(\sigma, b_1|\beta, \mathcal{B}, \phi, A \cup \{(b_1, \text{lbl}, s_1)\}, \mathcal{O})$;
- RIGHTARC$_{\text{lbl}}[(\sigma|s_2|s_1, \beta, \mathcal{B}, \phi, A, \mathcal{O})] =$ $(\sigma|s_2, \beta, \mathcal{B}, \phi, A \cup \{(s_2, \text{lbl}, s_1)\}, \mathcal{O})$;
- BUBBLEOPEN$_{\text{lbl}}[(\sigma|s_2|s_1, \beta, \mathcal{B}, \phi, A, \mathcal{O})] =$ $(\sigma|\alpha, \beta, \mathcal{B} \cup \{\alpha\}, \phi', A \cup \{(\alpha, \text{conj}, s_2), (\alpha, \text{lbl}, s_1)\}, \mathcal{O} \cup \{\alpha\})$, where α is a new bubble, and $\phi' = \phi \Cup \{\alpha \mapsto \psi(s_2) \cup \psi(s_1)\}$ (i.e., ϕ' is almost the same as ϕ, but with α added to the function's domain, mapped by the new function to cover the projections of both s_2 and s_1);
- BUBBLEATTACH$_{\text{lbl}}[(\sigma|s_2|s_1, \beta, \mathcal{B}, \phi, A, \mathcal{O})] =$ $(\sigma|s_2, \beta, \mathcal{B}, \phi', A \cup \{s_2, \text{lbl}, s_1\}, \mathcal{O})$, where $\phi' = \phi \Cup \{s_2 \mapsto \phi(s_2) \cup \psi(s_1)\}$;
- BUBBLECLOSE$[(\sigma|s_1, \beta, \mathcal{B}, \phi, A, \mathcal{O})] =$ $(\sigma, s_1|\beta, \mathcal{B}, \phi, A, \mathcal{O} \backslash \{s_1\})$.

3.2 Soundness and Completeness

In this section, we show that our Bubble-Hybrid transition system is both sound and complete (defined below) with respect to the subclass of projective bubble trees.[6]

Define a *valid* transition sequence $\pi = t_1, \ldots, t_m$ for a given sentence W to be a sequence such that for the corresponding sequence of configurations $c_0, \ldots, c_m$, we have $c_0 = c^i(W)$, $c_i = t_i(c_{i-1})$, and $c_m \in C_\tau$, We can then state soundness and completeness properties, and present high-level proof sketches below, adapted from Nivre's (2008) proof frameworks.

Lemma 1. (Soundness) *Every valid transition sequence π produces a projective bubble tree.*

Proof Sketch. We examine the requirements for a projective bubble tree one by one. The set of edges satisfies the tree constraints since every bubble except for the singleton bubble of RT must have an in-degree of one to have been reduced from the stack, and the topological order of reductions implies acyclicness. Lexical coverage is guaranteed by c^i. Roothood is safeguarded by the transition pre-conditions. Non-duplication is ensured because newly-created bubbles are strictly larger. All the other properties can be proved by induction over the lengths of transition sequence prefixes since each of our transitions preserves zero partial overlap, containment, and projectivity constraints. □

Lemma 2. (Completeness) *For every projective bubble tree over any given sentence W, there exists a corresponding valid transition sequence π.*

Proof Sketch. The proof proceeds by strong induction on sentence length. We omit relation labels without loss of generality. The base case of $|W| = 1$ is trivial. For the inductive step, we enumerate how to decompose the tree's top-level

[6] More precisely, our transition system handles the subset where each non-singleton bubble has ≥ 2 internal children.

structure. (1) When the root has multiple children: Due to projectivity, each child bubble tree τ_i covers a consecutive span of words $w_{x_i}, \ldots, w_{y_i}$ that are shorter than $|W|$. Based on the induction hypothesis, there exisits a valid transition sequence π_i to construct the child tree over $\mathsf{RT}, w_{x_i}, \ldots, w_{y_i}$. Here we let π_i to denote the transition sequence excluding the always-present final RIGHTARC transition that attaches the subtree to RT; this is for explicit illustration of what transitions to take once the subtrees are constructed. The full tree can be constructed by $\pi = \pi_1$, RIGHTARC, π_2, RIGHTARC, ... (expanding each π_i sequence into its component transitions), where we simply attach each subtree to RT immediately after it is constructed. (2) When the root has a single child bubble α, we cannot directly use the induction hypothesis since α covers the same number of words as W. Thus we need to further enumerate the top-level structure of α. (2a) If α has children with their projections outside of $\phi(\alpha)$, then we can find a sequence π_0 for constructing the shorter-length bubble α and placing it on the buffer (this corresponds to an empty transition sequence if α is a singleton; otherwise, π_0 ends with a BUBBLECLOSE transition) and π_is for α's outside children; say it has l children left of its contents. We construct the entire tree via $\pi = \pi_1,\ldots,\pi_l$, π_0, LEFTARC, ..., LEFTARC, SHIFT, π_{l+1}, RIGHTARC, ..., RIGHTARC, where we first construct all the left outside children and leave them on the stack, next build the bubble α and use LEFTARC transitions to attach its left children while it is on the buffer, then shift α to the stack before finally continuing on building its right children subtrees, each immediately followed by a RIGHTARC transition. (2b) If α is a non-singleton bubble without any outside children, but each of its inside children can be parsed through π_i based on the inductive hypothesis, then we can define $\pi = \pi_1,\pi_2$, BUBBLEOPEN, π_3, BUBBLEATTACH, ..., BUBBLECLOSE, SHIFT, RIGHTARC, where we use a BUBBLEOPEN transition once the first two bubble-internal children are built, each subsequent child is attached via BUBBLEATTACH immediately after construction, and the final three transitions ensure proper closing of the bubble and its attachment to RT. □

4 Models

Our model architecture largely follows that of Kiperwasser and Goldberg's (2016) neural Arc-Hybrid parser, but we additionally introduce feature composition for non-singleton bubbles, and a rescoring module to reduce frequent coordination-boundary prediction errors. Our model has five components: feature extraction, bubble-feature composition, transition scoring, label scoring, and boundary subtree rescoring.

Feature Extraction We first extract contextualized features for each token using a bidirectional LSTM (Graves and Schmidhuber, 2005):

$$[\mathbf{w}_0, \mathbf{w}_1, \ldots, \mathbf{w}_n] = \text{bi-LSTM}([\mathsf{RT}, w_1, \ldots, w_n]),$$

where the inputs to the bi-LSTM are concatenations of word embeddings, POS-tag embeddings, and character-level LSTM embeddings. We also report experiments replacing the bi-LSTM with pre-trained BERT features (Devlin et al., 2019).

Bubble-Feature Composition We initialize the features[7] for each singleton bubble $\mathcal{B}_i$ in the initial configuration to be $\mathbf{v}_{\mathcal{B}_i} = \mathbf{w}_i$. For a non-singleton bubble α, we use recursively composed features

$$\mathbf{v}_\alpha = g(\{\mathbf{v}_{\alpha'} | (\alpha, \text{conj}, \alpha') \in A\}),$$

where g is a composition function combining features from the co-heads (conjuncts) immediately inside the bubble.[8] For our model, for any $V' = \{\mathbf{v}_{i_1}, \ldots, \mathbf{v}_{i_N}\}$, we set

$$g(V') = \tanh(\mathbf{W}^g \cdot \text{mean}(V')),$$

where mean() computes element-wise averages and $\mathbf{W}^g$ is a learnable square matrix. We also experiment with a parameter-free version: $g = \text{mean}$. Neither of the feature functions distinguishes between open and closed bubbles, so we append to each $\mathbf{v}$ vector an indicator-feature embedding based on whether the bubble is open, closed, or singleton.

Transition Scoring Given the current parser configuration c, the model predicts the best unlabeled transition to take among all valid transitions valid(c) whose pre-conditions are satisfied. We

[7] We adopt the convenient abuse of notation of allowing indexing by arbitrary objects.

[8] Comparing with the subtree-feature composition functions in dependency parsing that are motivated by asymmetric headed constructions (Dyer et al., 2015; de Lhoneux et al., 2019; Basirat and Nivre, 2021), our definition focuses on composing features from an unordered set of vectors representing the conjuncts in a bubble. The composition function is recursively applied when there are nested bubbles.

model the log-linear probability of taking an action with a multi-layer perceptron (MLP):

$$P(t|c) \propto \exp(\text{MLP}_t^{\text{trans}}([\mathbf{v}_{s_3} \circ \mathbf{v}_{s_2} \circ \mathbf{v}_{s_1} \circ \mathbf{v}_{b_1}])),$$

where $\circ$ denotes vector concatenation, s_1 through s_3 are the first through third topmost items on the stack, and b_1 is the immediately accessible buffer item. We experiment with varying the number of stack items to extract features from.

Label Scoring We separate edge-label prediction from (unlabeled) transition prediction, but the scoring function takes a similar form:

$$P(l|c,t) \propto \exp(\text{MLP}_l^{\text{lbl}}([\mathbf{v}_{h(c,t)} \circ \mathbf{v}_{d(c,t)}])),$$

where $(h(c,t), l, d(c,t))$ is the edge to be added into the partial bubble tree in $t(c)$.

Boundary Subtree Rescoring In our preliminary error analysis, we find that our models tend to make more mistakes at the boundaries of full coordination phrases than at the internal conjunct boundaries, due to incorrect attachments of children choosing between the phrasal bubble and the first/last conjunct. For example, our initial model predicts "if you *owned it* and *liked it Friday*" instead of the annotated "if you *owned it* and *liked it* Friday" (the predicted and gold conjuncts are both italicized and underlined), incorrectly attaching "Friday" to "liked". We attribute this problem to the greedy nature of our first formulation of the parser, and propose to mitigate the issue through rescoring. To rescore boundary attachments of a non-singleton bubble α, for each of the left dependents d of α and its first conjunct α_f, we (re)-decide the attachment via

$$P(\alpha \rightarrow d|\alpha_f) = \text{logistic}(\text{MLP}^{\text{re}}([\mathbf{v}_d \circ \mathbf{v}_\alpha \circ \mathbf{v}_{\alpha_f}])),$$

and similarly for the last conjunct α_l and a potential right dependent.

Training and Inference Our parser is a locally-trained greedy parser. In training, we optimize the model parameters to maximize the log-likelihoods of predicting the target transitions and labels along the paths generating the gold bubble trees, and the log-likelihoods of the correct attachments in rescoring;[9] during inference, the parser greedily commits to the highest-scoring transition and label for each of its current parser configurations, and after reaching a terminal configuration, it rescores and readjusts all boundary subtree attachments.

[9] We leave the definition of dynamic oracles (Goldberg and Nivre, 2013) for bubble tree parsing to future work.

	Exact		Inner	
	All	NP	All	NP
FG16	–	–	72.70	76.10
TSM17	71.08	75.01	73.74	77.25
TSM19	75.47	77.83	77.74	80.06
Ours	76.48	81.63	78.30	84.03
+BERT	83.74	85.26	84.46	86.22

Table 2: F1 scores on the PTB test set. See Appendix C for precision, recall and dev set results.

	Exact	Whole
HSOM09	–	61.5
FG16	–	64.14
TSM17	55.22	66.31
TSM19	61.22	61.31
Ours	67.09	68.23
+BERT	79.18	80.41

Table 3: Recall results on the GENIA dataset (we report recall instead of F1 scores following prior work). See Appendix C for detailed results per constituent type.

5 Empirical Results

Task and Evaluation We validate the utility of our transition-based parser using the task of coordination structure prediction. Given an input sentence, the task is to identify all coordination structures and the spans for all their conjuncts within that sentence. We mainly evaluate based on *exact* metrics which count a prediction of a coordination structure as correct if and only if all of its conjunct spans are correct. To facilitate comparison with pre-existing systems that do not attempt to identify all conjunct boundaries, following Teranishi et al. (2017, 2019), we also consider *inner* (=only consider the correctness of the two conjuncts adjacent to the coordinator) and *whole* (=only consider the boundary of the whole coordinated phrase) metrics.

Data and Experimental Setup We experiment with two English datasets, the Penn Treebank (PTB; Marcus et al., 1993, newswire) with added coordination annotations (Ficler and Goldberg, 2016a) and the GENIA treebank (Kim et al., 2003, research abstracts). We use the conversion tool distributed with the Stanford Parser (Schuster and Manning, 2016) to extract UD trees from the PTB-style phrase-structure annotations, which we then merge with coordination annotations to form bub-

	Bubble-Hybrid (Ours)		Edge-Factored	
	Prec.	Rec.	Prec.	Rec.
punct	92.56	92.52	92.92	92.85
case	97.46	98.14	97.71	98.26
compound	94.12	95.18	94.24	95.02
det	98.85	99.13	98.70	99.06
nsubj	97.69	97.72	98.01	97.92
nmod	93.04	93.45	93.20	93.62
amod	94.52	93.43	94.61	93.95
...	...	...	...	...
conj	92.68	93.20	92.52	93.04
UAS	95.81		95.99	
LAS	94.46		94.56	

Table 4: PTB test-set results, comparing our transition-based bubble parser and an edge-factored graph-based parser, both using a BERT-based feature encoder. The relation labels are ordered by decreasing frequency. While our transition-based bubble parser slightly underperforms the graph-based dependency parser generally, perhaps due to the disadvantage of greedy decoding, it gives slightly better precision and recall on the "conj" relation type.

Rescoring	+	−
Ours (bi-LSTM)	77.10	76.27
• g = mean	75.51	74.16
• $\{\mathbf{v}_{s_2}, \mathbf{v}_{s_1}, \mathbf{v}_{b_1}\}$	76.05	74.87
• $\{\mathbf{v}_{s_1}, \mathbf{v}_{b_1}\}$	76.33	73.85
• $\{\mathbf{v}_{b_1}\}$	50.27	35.14
• +BERT	84.40	83.70

Table 5: Exact F1 scores of different model variations on the PTB dev set, w/ and w/o the rescoring module.

ble trees. We follow prior work in reporting PTB results on its standard splits and GENIA results using 5-fold cross-validation.[10] During training (but not test), we discard all non-projective sentences. See Appendix A for dataset pre-processing and statistics and Appendix B for implementation details.

Baseline Systems We compare our models with several baseline systems. Hara et al. (2009, HSOM09) use edit graphs to explicitly align coordinated conjuncts based on the idea that they are usually similar; Ficler and Goldberg (2016b, FG16) score candidate coordinations extracted from a phrase-structure parser by modeling their symmetry and replaceability properties; Teranishi et al. (2017, TSM17) directly predict boundaries of coordinated phrases and then split them into conjuncts;[11] Teranishi et al. (2019, TSM19) use separate neural models to score the inner and outer boundaries of conjuncts relative to the coordinators, and then use a chart parser to find the globally-optimal coordination structures.

Complexity	All	Simple	Complex
TSM17	66.09	72.90	50.37
TSM19	70.90	78.16	54.16
Ours	72.97	79.97	56.82
+BERT	80.07	83.74	71.59

Table 6: Per-sentence exact match on the PTB test set. *Simple* includes sentences with only one two-conjunct coordination, and *complex* contains the other cases.

Main Results Table 2 and Table 3 show the main evaluation results on the PTB and GENIA datasets. Our models surpass all prior results on both datasets. While the BERT improvements may not seem surprising, we note that Teranishi et al. (2019) report that their pre-trained language models — specifically, static ELMo embeddings — fail to improve their model performance.

General Parsing Results We also evaluate our models on standard parsing metrics by converting the predicted bubble trees to UD-style dependency trees. On PTB, our parsers reach unlabeled and labeled attachment scores (UAS/LAS) of 95.81/94.46 with BERT and 94.49/92.88 with bi-LSTM, which are similar to the scores of prior transition-based parsers equipped with similar feature extractors (Kiperwasser and Goldberg, 2016; Mohammadshahi and Henderson, 2020).[12] Table 4 compares the general parsing results of our bubble parser and an edge-factored graph-based dependency parser based on Dozat and Manning's (2017) parser architecture and the same feature encoder as our parser and trained on the same data. Our bubble parser shows a slight improvement on identifying the "conj" relations, despite having a lower overall accuracy due to the greedy nature of our transition-based decoder. Additionally, our

[10] We affirm that, as is best practice, only two test-set/crossval-suite runs occurred (one with BERT and one without), happening after we fixed everything else; that is, no other models were tried after seeing the first test-set/cross-validation results with and without BERT.

[11] We report results for the extended model of TSM17 as described by Teranishi et al. (2019).

[12] Results are not strictly comparable with previous PTB evaluations that mostly focus on non-UD dependency conversions. Table 4 makes a self-contained comparison using the same UD-based and coordination-merged data conversions.

bubble parser simultaneously predicts the boundaries of each coordinated phrase and conjuct, while a typical dependency parser cannot produce such structures.

Model Analysis Table 5 shows results of our models with alternative bubble-feature composition functions and varying feature-set sizes. We find that the parameterized form of composition function g performs better, and the F1 scores mostly degrade as we use fewer features from the stack. Interestingly, the importance of our rescoring module becomes more prominent when we use fewer features. Our results resonate with Shi et al.'s (2017) findings on Arc-Hybrid that we need at least one stack item but not necessarily two. Table 6 shows that our model performs better than previous methods on complex sentences with multiple coordination structures and/or more than two conjuncts, especially when we use BERT as feature extractor.

6 Related Work

Coordination Structure Prediction Very early work with heuristic, non-learning-based approaches (Agarwal and Boggess, 1992; Kurohashi and Nagao, 1994) typically report difficulties in distinguishing shared modifiers from private ones, although such heuristics have been recently incorporated in unsupervised work (Sawada et al., 2020). Generally, researchers have focused on symmetry principles, seeking to align conjuncts (Kurohashi and Nagao, 1994; Shimbo and Hara, 2007; Hara et al., 2009; Hanamoto et al., 2012), since coordinated conjuncts tend to be semantically and syntactically similar (Hogan, 2007), as attested to by psycholinguistic evidence of structural parallelism (Frazier et al., 1984, 2000; Dubey et al., 2005). Ficler and Goldberg (2016a) and Teranishi et al. (2017) additionally leverage the linguistic principle of replaceability — one can typically replace a coordinated phrase with one of its conjuncts without the sentence becoming incoherent; this idea has resulted in improved open information extraction (Saha and Mausam, 2018). Using these principles may further improve our parser.

Coordination in Constituency Grammar While our paper mainly focuses on enhancing dependency-based syntactic analysis with coordination structures, coordination is a well-studied topic in constituency-based syntax (Zhang, 2009), including proposals and treatments under lexical functional grammar (Kaplan and Maxwell III, 1988), tree-adjoining grammar (Sarkar and Joshi, 1996; Han and Sarkar, 2017), and combinatory categorial grammar (Steedman, 1996, 2000).

Tesnière Dependency Structure Sangati and Mazza (2009) propose a representation that is faithful to Tesnière's (1959) original framework. Similar to bubble trees, their structures include special attention to coordination structures respecting conjunct symmetry, but they also include constructs to handle other syntactic notions currently beyond our parser's scope.[13] Such representations have been used for re-ranking (Sangati, 2010), but not for (direct) parsing. Perhaps our work can inspire a future Tesnière Dependency Structure parser.

Non-constituent Coordination Seemingly incomplete (non-constituent) conjuncts are particularly challenging (Milward, 1994), and our bubble parser currently has no special mechanism for them. Dependency-based analyses have adapted by extending to a graph structure (Gerdes and Kahane, 2015) or explicitly representing elided elements (Schuster et al., 2017). It may be straightforward to integrate the latter into our parser, à la Kahane's (1997) proposal of phonologically-empty bubbles.

7 Conclusion

We revisit Kahane's (1997) bubble tree representations for explicitly encoding coordination boundaries as a viable alternative to existing mechanisms in dependency-based analysis of coordination structures. We introduce a transition system that is both sound and complete with respect to the subclass of projective bubble trees. Empirically, our bubble parsers achieve state-of-the-art results on the task of coordination structure prediction on two English datasets. Future work may extend the research scope to other languages, graph-based, and non-projective parsing methods.

Acknowledgements We thank the anonymous reviewers for their constructive comments, Yue Guo for discussion, and Hiroki Teranishi for help with experiment setup. This work was supported in part by a Bloomberg Data Science Ph.D. Fellowship to Tianze Shi and a gift from Bloomberg to Lillian Lee.

[13] For example, differentiating content and function words which has recently been explored by Basirat and Nivre (2021).

References

Rajeev Agarwal and Lois Boggess. 1992. A simple but useful approach to conjunct identification. In *Proceedings of the 30th Annual Meeting of the Association for Computational Linguistics*, pages 15–21, Newark, Delaware, USA. Association for Computational Linguistics.

Ali Basirat and Joakim Nivre. 2021. Syntactic nuclei in dependency parsing – A multilingual exploration. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 1376–1387, Online. Association for Computational Linguistics.

Gosse Bouma, Djamé Seddah, and Daniel Zeman. 2020. Overview of the IWPT 2020 shared task on parsing into enhanced Universal Dependencies. In *Proceedings of the 16th International Conference on Parsing Technologies and the IWPT 2020 Shared Task on Parsing into Enhanced Universal Dependencies*, pages 151–161, Online. Association for Computational Linguistics.

Michael Collins. 2003. Head-driven statistical models for natural language parsing. *Computational Linguistics*, 29(4):589–637.

Miryam de Lhoneux, Miguel Ballesteros, and Joakim Nivre. 2019. Recursive subtree composition in LSTM-based dependency parsing. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 1566–1576, Minneapolis, Minnesota, USA. Association for Computational Linguistics.

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

Timothy Dozat and Christopher D. Manning. 2017. Deep biaffine attention for neural dependency parsing. In *Proceedings of the 5th International Conference on Learning Representations*, Toulon, France. OpenReview.net.

Amit Dubey, Patrick Sturt, and Frank Keller. 2005. Parallelism in coordination as an instance of syntactic priming: Evidence from corpus-based modeling. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 827–834, Vancouver, British Columbia, Canada. Association for Computational Linguistics.

Chris Dyer, Miguel Ballesteros, Wang Ling, Austin Matthews, and Noah A. Smith. 2015. Transition-based dependency parsing with stack long short-term memory. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 334–343, Beijing, China. Association for Computational Linguistics.

Jessica Ficler and Yoav Goldberg. 2016a. Coordination annotation extension in the Penn tree bank. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 834–842, Berlin, Germany. Association for Computational Linguistics.

Jessica Ficler and Yoav Goldberg. 2016b. A neural network for coordination boundary prediction. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 23–32, Austin, Texas, USA. Association for Computational Linguistics.

Jessica Ficler and Yoav Goldberg. 2017. Improving a strong neural parser with conjunction-specific features. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 343–348, Valencia, Spain. Association for Computational Linguistics.

Lyn Frazier, Alan Munn, and Charles Clifton. 2000. Processing coordinate structures. *Journal of Psycholinguistic Research*, 29(4):343–370.

Lyn Frazier, Lori Taft, Tom Roeper, Charles Clifton, and Kate Ehrlich. 1984. Parallel structure: A source of facilitation in sentence comprehension. *Memory & Cognition*, 12(5):421–430.

Kim Gerdes and Sylvain Kahane. 2015. Non-constituent coordination and other coordinative constructions as dependency graphs. In *Proceedings of the Third International Conference on Dependency Linguistics (Depling 2015)*, pages 101–110, Uppsala, Sweden. Uppsala University.

Aleksej V. Gladkij. 1968. "On describing the syntactic structure of a sentence" (in Russian with English summary). *Computational Linguistics*, 7:21–44.

Yoav Goldberg and Michael Elhadad. 2010. Inspecting the structural biases of dependency parsing algorithms. In *Proceedings of the Fourteenth Conference on Computational Natural Language Learning*, pages 234–242, Uppsala, Sweden. Association for Computational Linguistics.

Yoav Goldberg and Joakim Nivre. 2013. Training deterministic parsers with non-deterministic oracles. *Transactions of the Association for Computational Linguistics*, 1:403–414.

Alex Graves and Jürgen Schmidhuber. 2005. Framewise phoneme classification with bidirectional LSTM and other neural network architectures. *Neural Networks*, 18(5):602–610.

Stefan Grünewald, Prisca Piccirilli, and Annemarie Friedrich. 2021. Coordinate constructions in English enhanced Universal Dependencies: Analysis and computational modeling. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 795–809, Online. Association for Computational Linguistics.

Jan Hajič, Eduard Bejček, Jaroslava Hlavacova, Marie Mikulová, Milan Straka, Jan Štěpánek, and Barbora Štěpánková. 2020. Prague dependency treebank - consolidated 1.0. In *Proceedings of the 12th Language Resources and Evaluation Conference*, pages 5208–5218, Marseille, France. European Language Resources Association.

Jan Hajič, Barbora Vidová Hladká, Jarmila Panevová, Eva Hajičová, Petr Sgall, and Petr Pajas. 2001. Prague dependency treebank 1.0 (LDC2001T10).

Chung-hye Han and Anoop Sarkar. 2017. Coordination in TAG without the Conjoin Operation. In *Proceedings of the 13th International Workshop on Tree Adjoining Grammars and Related Formalisms*, pages 43–52, Umeå, Sweden. Association for Computational Linguistics.

Atsushi Hanamoto, Takuya Matsuzaki, and Jun'ichi Tsujii. 2012. Coordination structure analysis using dual decomposition. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 430–438, Avignon, France. Association for Computational Linguistics.

Kazuo Hara, Masashi Shimbo, Hideharu Okuma, and Yuji Matsumoto. 2009. Coordinate structure analysis with global structural constraints and alignment-based local features. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 967–975, Suntec, Singapore. Association for Computational Linguistics.

Deirdre Hogan. 2007. Coordinate noun phrase disambiguation in a generative parsing model. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 680–687, Prague, Czech Republic. Association for Computational Linguistics.

Richard A. Hudson. 1984. *Word Grammar*. Blackwell, Oxford.

Timo Järvinen and Pasi Tapanainen. 1998. Towards an implementable dependency grammar. In *Processing of Dependency-Based Grammars*, pages 1–10, Montreal, Quebec, Canada. Association for Computational Linguistics.

Richard Johansson and Pierre Nugues. 2007. Extended constituent-to-dependency conversion for English. In *Proceedings of the 16th Nordic Conference of Computational Linguistics (NODALIDA 2007)*, pages 105–112, Tartu, Estonia. University of Tartu, Estonia.

Sylvain Kahane. 1997. Bubble trees and syntactic representations. In *Proceedings of the 5th Meeting of Mathematics of Language*, pages 70–76, Saarbrücken, Germany.

Ronald M. Kaplan and John T. Maxwell III. 1988. Constituent coordination in lexical-functional grammar. In *Proceedings of International Conference on Computational Linguistics*, pages 303–305, Budapest, Hungary.

Jin Dong Kim, Tomoko Ohta, Yuka Tateisi, and Jun'ichi Tsujii. 2003. GENIA corpus—a semantically annotated corpus for bio-textmining. *Bioinformatics*, 19(suppl_1):i180–i182.

Eliyahu Kiperwasser and Yoav Goldberg. 2016. Simple and accurate dependency parsing using bidirectional LSTM feature representations. *Transactions of the Association for Computational Linguistics*, 4:313–327.

Sandra Kübler, Ryan McDonald, and Joakim Nivre. 2009. *Dependency parsing*. Synthesis Lectures on Human Language Technologies. Morgan & Claypool Publishers.

Marco Kuhlmann, Carlos Gómez-Rodríguez, and Giorgio Satta. 2011. Dynamic programming algorithms for transition-based dependency parsers. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 673–682, Portland, Oregon, USA. Association for Computational Linguistics.

Sadao Kurohashi and Makoto Nagao. 1994. A syntactic analysis method of long Japanese sentences based on the detection of conjunctive structures. *Computational Linguistics*, 20(4):507–534.

Vincenzo Lombardo and Leonardo Lesmo. 1998. Unit coordination and gapping in dependency theory. In *Processing of Dependency-Based Grammars*, pages 11–20, Montreal, Quebec, Canada. Association for Computational Linguistics.

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

Igor Mel'čuk. 1988. *Dependency Syntax: Theory and Practice*. State University Press of New York, Albany.

Igor Mel'čuk. 2003. Levels of dependency in linguistic description: Concepts and problems. In Vilmos Ágel, Ludwig M. Eichinger, Hans Werner Eroms,

Peter Hellwig, Hans Jürgen Heringer, and Henning Lobin, editors, *Dependency and Valency: An International Handbook of Contemporary Research*, volume 1, pages 188–229. Walter de Gruyter, Berlin.

David Milward. 1994. Non-constituent coordination: Theory and practice. In *Proceedings of the 15th International Conference on Computational Linguistics*, volume 2, pages 935–941, Kyoto, Japan.

Alireza Mohammadshahi and James Henderson. 2020. Graph-to-graph Transformer for transition-based dependency parsing. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 3278–3289, Online. Association for Computational Linguistics.

Jens Nilsson, Joakim Nivre, and Johan Hall. 2006. Graph transformations in data-driven dependency parsing. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 257–264, Sydney, Australia. Association for Computational Linguistics.

Joakim Nivre. 2008. Algorithms for deterministic incremental dependency parsing. *Computational Linguistics*, 34(4):513–553.

Joakim Nivre, Marie-Catherine de Marneffe, Filip Ginter, Yoav Goldberg, Jan Hajič, Christopher D. Manning, Ryan McDonald, Slav Petrov, Sampo Pyysalo, Natalia Silveira, Reut Tsarfaty, and Daniel Zeman. 2016. Universal Dependencies v1: A multilingual treebank collection. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pages 1659–1666, Portorož, Slovenia. European Language Resources Association.

Joakim Nivre, Marie-Catherine de Marneffe, Filip Ginter, Jan Hajič, Christopher D. Manning, Sampo Pyysalo, Sebastian Schuster, Francis Tyers, and Daniel Zeman. 2020. Universal Dependencies v2: An evergrowing multilingual treebank collection. In *Proceedings of the 12th Language Resources and Evaluation Conference*, pages 4034–4043, Marseille, France. European Language Resources Association.

Joakim Nivre, Paola Marongiu, Filip Ginter, Jenna Kanerva, Simonetta Montemagni, Sebastian Schuster, and Maria Simi. 2018. Enhancing Universal Dependency treebanks: A case study. In *Proceedings of the Second Workshop on Universal Dependencies (UDW 2018)*, pages 102–107, Brussels, Belgium. Association for Computational Linguistics.

Martin Popel, David Mareček, Jan Štěpánek, Daniel Zeman, and Zděněk Žabokrtský. 2013. Coordination structures in dependency treebanks. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 517–527, Sofia, Bulgaria. Association for Computational Linguistics.

Owen Rambow. 2010. The simple truth about dependency and phrase structure representations: An opinion piece. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 337–340, Los Angeles, California, USA. Association for Computational Linguistics.

Ines Rehbein, Julius Steen, Bich-Ngoc Do, and Anette Frank. 2017. Universal Dependencies are hard to parse — or are they? In *Proceedings of the Fourth International Conference on Dependency Linguistics (Depling 2017)*, pages 218–228, Pisa, Italy. Linköping University Electronic Press.

Swarnadeep Saha and Mausam. 2018. Open information extraction from conjunctive sentences. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 2288–2299, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Swarnadeep Saha, Harinder Pal, and Mausam. 2017. Bootstrapping for numerical Open IE. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 317–323, Vancouver, Canada. Association for Computational Linguistics.

Federico Sangati. 2010. A probabilistic generative model for an intermediate constituency-dependency representation. In *Proceedings of the ACL 2010 Student Research Workshop*, pages 19–24, Uppsala, Sweden. Association for Computational Linguistics.

Federico Sangati and Chiara Mazza. 2009. An English dependency treebank à la Tesnière. In *Proceedings of the 8th International Workshop on Treebanks and Linguistic Theories*, pages 173–184, Milan, Italy.

Anoop Sarkar and Aravind Joshi. 1996. Coordination in tree adjoining grammars: Formalization and implementation. In *Proceedings of the 16th International Conference on Computational Linguistics*, pages 610–615, Copenhagen, Denmark.

Yuya Sawada, Takashi Wada, Takayoshi Shibahara, Hiroki Teranishi, Shuhei Kondo, Hiroyuki Shindo, Taro Watanabe, and Yuji Matsumoto. 2020. Coordination boundary identification without labeled data for compound terms disambiguation. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 3043–3049, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Sebastian Schuster, Matthew Lamm, and Christopher D. Manning. 2017. Gapping constructions in Universal Dependencies v2. In *Proceedings of the NoDaLiDa 2017 Workshop on Universal Dependencies (UDW 2017)*, pages 123–132, Gothenburg, Sweden. Association for Computational Linguistics.

Sebastian Schuster and Christopher D. Manning. 2016. Enhanced English Universal Dependencies: An im-

proved representation for natural language understanding tasks. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*, pages 2371–2378, Portorož, Slovenia. European Language Resources Association.

Tianze Shi, Liang Huang, and Lillian Lee. 2017. Fast(er) exact decoding and global training for transition-based dependency parsing via a minimal feature set. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 12–23, Copenhagen, Denmark. Association for Computational Linguistics.

Masashi Shimbo and Kazuo Hara. 2007. A discriminative learning model for coordinate conjunctions. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 610–619, Prague, Czech Republic. Association for Computational Linguistics.

Mark Steedman. 1996. *Surface Structure and Interpretation*. Number 30 in Linguistic Inquiry Monograph. The MIT Press, Cambridge, Massachusetts.

Mark Steedman. 2000. *The Syntactic Process*. MIT Press, Cambridge, MA, USA.

Hiroki Teranishi, Hiroyuki Shindo, and Yuji Matsumoto. 2017. Coordination boundary identification with similarity and replaceability. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 264–272, Taipei, Taiwan. Asian Federation of Natural Language Processing.

Hiroki Teranishi, Hiroyuki Shindo, and Yuji Matsumoto. 2019. Decomposed local models for coordinate structure parsing. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 3394–3403, Minneapolis, Minnesota, USA. Association for Computational Linguistics.

Lucien Tesnière. 1959. *Eléments de Syntaxe Structurale*. Librairie C. Klincksieck, Paris.

Niina Ning Zhang. 2009. *Coordination in Syntax*. Cambridge Studies in Linguistics. Cambridge University Press, New York.

Appendix A Dataset Processing and Statistics

We follow Teranishi et al. (2019) and use the same dataset splits and pre-processing steps. For the Penn Treebank (PTB; Marcus et al., 1993) data with added coordination annotations (Ficler and Goldberg, 2016a), we use WSJ sections 02-21 for training, section 22 for development, and section 23 for test sets respectively. We also use Teranishi et al.'s (2019) pre-processing steps in stripping quotation marks surrounding PTB coordinated phrases to normalize irregular coordinations. This results in 39,832/1,700/2,416 sentences and 19,890/848/1,099 coordination structures in train/dev/test splits respectively. For the GENIA treebank (Kim et al., 2003), we use the beta version of the corpus and follow the same 5-fold cross-validation splits as Teranishi et al. (2019). In total, GENIA contains 2,508 sentences and 3,598 coordination structures.

To derive bubble tree representations, we first convert the PTB-style phrase-structure trees in both treebanks with the conversion tool (Schuster and Manning, 2016) provided by the Stanford CoreNLP toolkit version 4.2.0 into Universal Dependencies (UD; Nivre et al., 2016) style. We then merge the UD trees with the bubbles formed by the coordination boundaries. We define the boundaries to be from the beginning of the first conjunct to the end of the last conjunct for each coordinated phrase. We attach all conjuncts to their corresponding bubbles with a "conj" label, and map any "conj"-labeled dependencies outside an annotated coordination to "dep". We resolve modifier scope ambiguities according to conjunct annotations: if the modifier is within the span of a conjunct, then it is a private modifier; otherwise, it is a shared modifier to the entire coordinated phrase and we attach it to the phrasal bubble. Since our transition system targets projective bubble trees, we filter out any non-projective trees during training (but still evaluate on them during testing). We retain 39,678 sentences, or 99.6% of the PTB training set, and 2,429 sentences, or 96.9% of the GENIA dataset.

Appendix B Implementation Details

Our implementation (https://www.github.com/tzshi/bubble-parser-acl21) is based on PyTorch.

We train our models by using the Adam optimizer (Kingma and Ba, 2015). After a fixed number of optimization steps (3,200 steps for PTB and 800 steps for GENIA, based on their training set sizes), we perform an evaluation on the dev set. If the dev set performance fails to improve within 5 consecutive evaluation rounds, we multiply the learning rate by 0.1. We terminate model training when the learning rate has dropped three times, and select the best model checkpoint based on dev set F1 scores according to the "exact" metrics.[14] For the BERT feature extractor, we finetune the pretrained case-sensitive $\text{BERT}_{\text{base}}$ model through the `transformers` package.[15] For the non-BERT model, we use pre-trained GloVe embeddings (Pennington et al., 2014).

Adam Optimizer:	
Initial learning rate for bi-LSTM	10^{-3}
Initial learning rate for BERT	10^{-5}
β_1	0.9
β_2	0.999
ϵ	10^{-8}
Minibatch size	8
Linear warmup steps	800
Gradient clipping L_2 norm	5.0
Inputs to bi-LSTM:	
Word-embedding dimensionality	100
POS tag-embedding dimensionality	32
Character bi-LSTM layers	1
Character bi-LSTM dimensionality	128
Bi-LSTM:	
Number of layers	3
Dimensionality	800
Dropout	0.3
MLPs (same for all MLPs):	
Number of hidden layers	1
Hidden layer dimensionality	400
Activation function	ReLU
Dropout	0.3

Table A1: Hyperparameters of our models.

Following prior practice, we embed gold POS tags as input features when using bi-LSTM for the models trained on the GENIA dataset, but we omit the POS tag embeddings for the PTB dataset.

The training process for each model takes roughly 10 hours using an RTX 2080 Ti GPU; model inference speed is 41.9 sentences per second.[16]

We select our hyperparameters by hand. Due to computational constraints, our hand-tuning has been limited to setting the dropout rates, and from the candidates set of $\{0.0, 0.1, 0.3, 0.5\}$ we chose

[14] Even though we report recall on GENIA, model selection is still performed using F1.

[15] github.com/huggingface/transformers

[16] We have not yet done extensive optimization regarding GPU batching for greedy transition-based parsers.

0.3 based on dev-set performance. Our hyperparameters are listed in Table A1.

Appendix C Extended Results

Table A2 and Table A3 include detailed evaluation results on the PTB and GENIA datasets.

References

Jessica Ficler and Yoav Goldberg. 2016a. Coordination annotation extension in the Penn tree bank. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 834–842, Berlin, Germany. Association for Computational Linguistics.

Jessica Ficler and Yoav Goldberg. 2016b. A neural network for coordination boundary prediction. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 23–32, Austin, Texas, USA. Association for Computational Linguistics.

Kazuo Hara, Masashi Shimbo, Hideharu Okuma, and Yuji Matsumoto. 2009. Coordinate structure analysis with global structural constraints and alignment-based local features. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 967–975, Suntec, Singapore. Association for Computational Linguistics.

Jin Dong Kim, Tomoko Ohta, Yuka Tateisi, and Jun'ichi Tsujii. 2003. GENIA corpus—a semantically annotated corpus for bio-textmining. *Bioinformatics*, 19(suppl_1):i180–i182.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *Proceedings of the 3rd International Conference on Learning Representations*, San Diego, California, USA.

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

Joakim Nivre, Marie-Catherine de Marneffe, Filip Ginter, Yoav Goldberg, Jan Hajič, Christopher D. Manning, Ryan McDonald, Slav Petrov, Sampo Pyysalo, Natalia Silveira, Reut Tsarfaty, and Daniel Zeman. 2016. Universal Dependencies v1: A multilingual treebank collection. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pages 1659–1666, Portorož, Slovenia. European Language Resources Association.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar. Association for Computational Linguistics.

Sebastian Schuster and Christopher D. Manning. 2016. Enhanced English Universal Dependencies: An improved representation for natural language understanding tasks. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*, pages 2371–2378, Portorož, Slovenia. European Language Resources Association.

Hiroki Teranishi, Hiroyuki Shindo, and Yuji Matsumoto. 2017. Coordination boundary identification with similarity and replaceability. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 264–272, Taipei, Taiwan. Asian Federation of Natural Language Processing.

Hiroki Teranishi, Hiroyuki Shindo, and Yuji Matsumoto. 2019. Decomposed local models for coordinate structure parsing. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 3394–3403, Minneapolis, Minnesota, USA. Association for Computational Linguistics.

		Dev						Test					
		All			NP			All			NP		
		P	R	F	P	R	F	P	R	F	P	R	F
Exact	TSM17	74.13	73.34	73.74	76.21	75.51	75.86	71.48	70.70	71.08	75.20	74.84	75.01
	TSM19	76.95	76.76	76.85	78.11	77.57	77.84	75.33	75.61	75.47	77.95	77.70	77.83
	Ours	77.19	77.00	77.10	80.00	79.63	79.82	76.62	76.34	76.48	81.89	81.37	81.63
	+BERT	84.25	84.55	84.40	88.53	88.33	88.43	83.85	83.62	83.74	85.33	85.19	85.26
Inner	FG16	72.34	72.25	72.29	75.17	74.82	74.99	72.81	72.61	72.70	76.91	75.31	76.10
	TSM17	76.04	75.23	75.63	77.82	77.11	77.47	74.14	73.33	73.74	77.44	77.07	77.25
	TSM19	79.19	79.00	79.10	80.64	80.09	80.36	77.60	77.88	77.74	80.19	79.93	80.06
	Ours	78.61	78.42	78.51	82.07	81.69	81.88	78.45	78.16	78.30	84.29	83.76	84.03
	+BERT	85.19	85.50	85.34	89.45	89.24	89.35	84.58	84.35	84.46	86.28	86.15	86.22

Table A2: Precision, recall, and F1 scores on the PTB dev and test sets.

		NP	VP	ADJP	S	PP	UCP	SBAR	ADVP	Others	All
	Count	2,317	465	321	188	167	60	56	21	3	3,598
Exact	TSM17	57.14	54.83	72.27	8.51	55.68	28.33	57.14	85.71	0.00	55.22
	TSM19	59.21	64.94	78.19	53.19	55.68	48.33	66.07	90.47	0.00	61.22
	Ours	68.28	58.71	86.29	56.38	55.09	51.67	58.93	95.24	0.00	67.09
	+BERT	79.41	76.34	88.79	77.13	73.05	61.67	76.79	100.0	33.33	79.18
Whole	HSOM09	64.2	54.2	80.4	22.9	59.9	36.7	51.8	85.7	66.7	61.5
	FG16	65.08	71.82	74.76	17.02	56.28	51.67	91.07	80.95	33.33	64.14
	TSM17	67.19	63.65	76.63	53.19	61.67	35.00	78.57	85.71	33.33	66.31
	TSM19	59.30	65.16	78.19	53.19	55.68	48.33	66.07	90.47	0.00	61.31
	Ours	69.40	59.35	87.85	57.45	56.89	51.67	62.50	95.24	0.00	68.23
	+BERT	80.58	76.77	90.03	78.19	76.65	61.67	82.14	100.0	33.33	80.41

Table A3: Recall on the GENIA dataset.