The Limitations of Limited Context for Constituency Parsing

Yuchen Li
Carnegie Mellon University
yuchenl4@andrew.cmu.edu

Andrej Risteski
Carnegie Mellon University
aristesk@andrew.cmu.edu

Abstract

Incorporating syntax into neural approaches in NLP has a multitude of practical and scientific benefits. For instance, a language model that is syntax-aware is likely to be able to produce better samples; even a discriminative model like BERT with a syntax module could be used for core NLP tasks like unsupervised syntactic parsing. Rapid progress in recent years was arguably spurred on by the empirical success of the Parsing-Reading-Predict architecture of (Shen et al., 2018a), later simplified by the Order Neuron LSTM of (Shen et al., 2019). Most notably, this is the first time neural approaches were able to successfully perform unsupervised syntactic parsing (evaluated by various metrics like F-1 score).

However, even heuristic (much less fully mathematical) understanding of why and when these architectures work is lagging severely behind. In this work, we answer representational questions raised by the architectures in (Shen et al., 2018a, 2019), as well as some transition-based syntax-aware language models (Dyer et al., 2016): *what kind of syntactic structure can current neural approaches to syntax represent*? Concretely, we ground this question in the sandbox of probabilistic context-free-grammars (PCFGs), and identify a key aspect of the representational power of these approaches: the *amount* and *directionality* of context that the predictor has access to when forced to make parsing decision. We show that with *limited context* (either bounded, or unidirectional), there are PCFGs, for which these approaches cannot represent the max-likelihood parse; conversely, if the context is *unlimited*, they can represent the max-likelihood parse of any PCFG.

1 Introduction

Neural approaches have been steadily making their way to NLP in recent years. By and large however, the neural techniques that have been scaled-up the most and receive widespread usage do not explicitly try to encode discrete structure that is natural to language, e.g. syntax. The reason for this is perhaps not surprising: neural models have largely achieved substantial improvements in *unsupervised* settings, BERT (Devlin et al., 2019) being the defacto method for unsupervised pre-training in most NLP settings. On the other hand unsupervised *syntactic* tasks, e.g. unsupervised syntactic parsing, have long been known to be very difficult tasks (Htut et al., 2018). However, since incorporating syntax has been shown to improve language modeling (Kim et al., 2019b) as well as natural language inference (Chen et al., 2017; Pang et al., 2019; He et al., 2020), syntactic parsing remains important even in the current era when large pre-trained models, like BERT (Devlin et al., 2019), are available.

Arguably, the breakthrough works in unsupervised constituency parsing in a neural manner were (Shen et al., 2018a, 2019), achieving F1 scores 42.8 and 49.4 on the WSJ Penn Treebank dataset (Htut et al., 2018; Shen et al., 2019). Both of these architectures, however (especially Shen et al., 2018a) are quite intricate, and it's difficult to evaluate what their representational power is (i.e. what *kinds* of structure can they recover). Moreover, as subsequent more thorough evaluations show (Kim et al., 2019b,a), these methods still have a rather large performance gap with the oracle binary tree (which is the best binary parse tree according to F1-score) — raising the question of *what is missing* in these methods.

We theoretically answer both questions raised in the prior paragraph. We quantify the representational power of two major frameworks in neural approaches to syntax: *learning a syntactic distance* (Shen et al., 2018a,b, 2019) and *learning to parse through sequential transitions* (Dyer et al., 2016; Chelba, 1997). To formalize our results, we con-

Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, pages 2675–2687
August 1–6, 2021.

sider the well-established sandbox of *probabilistic context-free grammars* (PCFGs). Namely, we ask:

When is a neural model based on a syntactic distance or transitions able to represent the max-likelihood parse of a sentence generated from a PCFG?

We focus on a crucial "hyperparameter" common to practical implementations of both families of methods that turns out to govern the representational power: the amount and type of context the model is allowed to use when making its predictions. Briefly, for every position t in the sentence, syntactic distance models learn a distance d_t to the previous token — the tree is then inferred from this distance; transition-based models iteratively construct the parse tree by deciding, at each position t, what operations to perform on a partial parse up to token t. A salient feature of both is the *context*, that is, *which tokens is d_t a function of* (correspondingly, which tokens can the choice of operations at token t depend on)?

We show that when the context is either *bounded* (that is, d_t only depends on a bounded window around the t-th token) or *unidirectional* (that is, d_t only considers the tokens to the left of the t-th token), there are PCFGs for which no distance metric (correspondingly, no algorithm to choose the sequence of transitions) works. On the other hand, if the context is *unbounded in both directions* then both methods work: that is, for any parse, we can design a distance metric (correspondingly, a sequence of transitions) that recovers it.

This is of considerable importance: in practical implementations the context is either bounded (e.g. in Shen et al., 2018a, the distance metric is parametrized by a convolutional kernel with a constant width) or unidirectional (e.g. in Shen et al., 2019, the distance metric is computed by a LSTM, which performs a left-to-right computation).

This formally confirms a conjecture of Htut et al. (2018), who suggested that because these models commit to parsing decision in a left-to-right fashion and are trained as a part of a language model, it may be difficult for them to capture sufficiently complex syntactic dependencies. Our techniques are fairly generic and seem amenable to analyzing other approaches to syntax. Finally, while the existence of *a particular* PCFG that is problematic for these methods doesn't necessarily imply that the difficulties will carry over to real-life data, the PCFGs that are used in our proofs closely track linguistic intuitions about difficult syntactic structures to infer: the parse depends on words that come much later in the sentence.

2 Overview of Results

We consider several neural architectures that have shown success in various syntactic tasks, most notably unsupervised constituency parsing and syntax-aware language modeling. The general framework these architectures fall under is as follows: to parse a sentence $W = w_1 w_2 ... w_n$ with a trained neural model, the sentence W is input into the model, which outputs o_t at each step t, and finally all the outputs $\{o_t\}_{t=1}^{n}$ are utilized to produce the parse.

Given unbounded time and space resources, by a seminal result of Siegelmann and Sontag (1992), an RNN implementation of this framework is Turing complete. In practice it is common to restrict the form of the output o_t in some way. In this paper, we consider the two most common approaches, in which o_t is a *real number representing a syntactic distance* (Section 2.1) (Shen et al., 2018a,b, 2019) or a *sequence of parsing operations* (Section 2.2) (Chelba, 1997; Chelba and Jelinek, 2000; Dyer et al., 2016). We proceed to describe our results for each architecture in turn.

2.1 Syntactic distance

Syntactic distance-based neural parsers train a neural network to learn a distance for each pair of adjacent words, depending on the context surrounding the pair of words under consideration. The distances are then used to induce a tree structure (Shen et al., 2018a,b).

For a sentence $W = w_1 w_2 ... w_n$, the syntactic distance between w_{t-1} and w_t ($2 \leq t \leq n$) is defined as $d_t = d(w_{t-1}, w_t \,|\, c_t)$, where c_t is the context that d_t takes into consideration [1]. We will show that restricting the surrounding context either in directionality, or in size, results in a poor representational power, while full context confers essentially perfect representational power with respect to PCFGs.

Concretely, if the context is full, we show:

Theorem (Informal, full context)**.** *For sentence W generated by any PCFG, if the computation of d_t has as context the full sentence and the position index under consideration, i.e. $c_t = (W, t)$ and*

[1] Note that this is not a conditional distribution—we use this notation for convenience.

$d_t = d(w_{t-1}, w_t \mid c_t)$*, then* d_t *can induce the maximum likelihood parse of* W.

On the flipside, if the context is unidirectional (i.e. unbounded left-context from the start of the sentence, and even possibly with a bounded look-ahead), the representational power becomes severely impoverished:

Theorem (Informal, limitation of left-to-right parsing via syntactic distance)**.** *There exists a PCFG* G *such that for any distance measure* d_t *whose computation incorporates only bounded context in at least one direction (left or right), e.g.*

$$c_t = (w_0, w_1, ..., w_{t+L'})$$
$$d_t = d(w_{t-1}, w_t \mid c_t)$$

the probability that d_t *induces the max likelihood parse is arbitrarily low.*

In practice, for computational efficiency, parametrizations of syntactic distances fall into the above assumptions of restricted context (Shen et al., 2018a). This puts the ability of these models to learn a complex PCFG syntax into considerable doubt. For formal definitions, see Section 4.2. For formal theorem statements and proofs, see Section 5.

Subsequently we consider ON-LSTM, an architecture proposed by Shen et al. (2019) improving their previous work (Shen et al., 2018a), which also is based on learning a syntactic distance, but in (Shen et al., 2019) the distances are reduced from the values of a carefully structured master forget gate (see Section 6). While we show ON-LSTM can *in principle* losslessly represent any parse tree (Theorem 3), calculating the gate values in a left to right fashion (as is done in practice) is subject to the same limitations as the syntactic distance approach:

Theorem (Informal, limitation of syntactic distance estimation based on ON-LSTM)**.** *There exists a PCFG* G *for which the probability that the syntactic distance converted from an ON-LSTM induces the max likelihood parse is arbitrarily low.*

For a formal statement, see Section 6 and in particular Theorem 4.

2.2 Transition-based parsing

In principle, the output o_t at each position t of a left-to-right neural models for syntactic parsing need not be restricted to a real-numbered distance or a carefully structured vector. It can also be a combinatorial structure — e.g. a sequence of *transitions* (Chelba, 1997; Chelba and Jelinek, 2000; Dyer et al., 2016). We adopt a simplification of the neural parameterization in (Dyer et al., 2016) (see Definition 4.7).

With full context, Dyer et al. (2016) describes an algorithm to find a sequence of transitions to represent any parse tree, via a "depth-first, left-to-right traversal" of the tree. On the other hand, without full context, we prove that transition-based parsing suffers from the same limitations:

Theorem (Informal, limitation of transition-based parsing without full context)**.** *There exists a PCFG* G*, such that for any learned transition-based parser with bounded context in at least one direction (left or right), the probability that it returns the max likelihood parse is arbitrarily low.*

For a formal statement, see Section 7, and in particular Theorem 5.

Remark. There is no immediate connection between the syntactic distance-based approaches (including ON-LSTM) and the transition-based parsing framework, so the limitations of transition-based parsing does not directly imply the stated negative results for syntactic distance or ON-LSTM, and vice versa.

2.3 The counterexample family

Most of our theorems proving limitations on bounded and unidirectional context are based on a PCFG family (Definition 2.1) which draws inspirations from natural language already suggested in (Htut et al., 2018): later words in a sentence can force different syntactic structures earlier in the sentence. For example, consider the two sentences: "*I drink coffee with milk.*" and "*I drink coffee with friends.*" Their only difference occurs at their very last words, but their parses differ at some earlier words in each sentence, too, as shown in Figure 1.

To formalize this intuition, we define the following PCFG.

Definition 2.1 (Right-influenced PCFG)**.** Let $m \geq 2, L' \geq 1$ be positive integers. The grammar $G_{m,L'}$ has starting symbol S, other non-terminals

$$A_k, B_k, A_k^l, A_k^r, B_k' \text{ for all } k \in \{1, 2, ..., m\},$$

and terminals

$$a_i \text{ for all } i \in \{1, 2, ..., m+1+L'\},$$

$$c_j \text{ for all } j \in \{1, 2, ..., m\}.$$

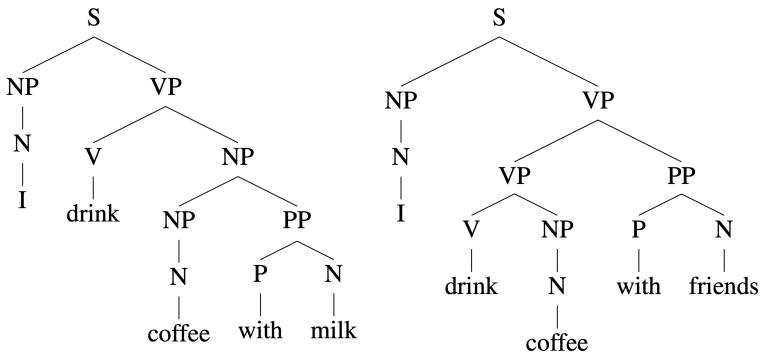


Figure 1: The parse trees of the two sentences: "*I drink coffee with milk.*" and "*I drink coffee with friends.*". Their only difference occurs at their very last words, but their parses differ at some earlier words in each sentence

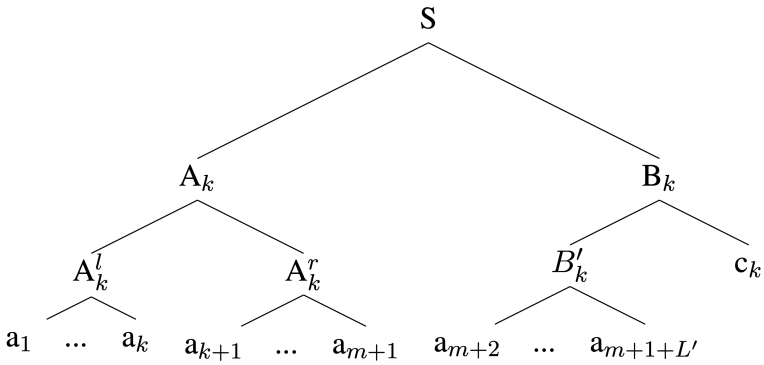


Figure 2: The structure of the parse tree of string $l_k = a_1 a_2 ... a_{m+1+L'} c_k \in L(G_{m,L'})$. Note that any l_{k_1} and l_{k_2} are almost the same except for the last token: the prefix $a_1 a_2 ... a_{m+1+L'}$ is shared among all strings in $L(G_{m,L'})$. However, their parses differ with respect to where A_k is split. The last token c_k is unique to l_k and hence determines the correct parse according to $G_{m,L'}$.

The rules of the grammar are

$$
\begin{aligned}
S &\to A_k B_k, \forall k \in \{1, 2, \ldots, m\} \text{ w. prob.} 1/m \\
A_k &\to A_k^l A_k^r \quad \text{w. prob. 1} \\
A_k^l &\to^* a_1 a_2 ... a_k \quad \text{w. prob. 1} \\
A_k^r &\to^* a_{k+1} a_{k+2} ... a_{m+1} \quad \text{w. prob. 1} \\
B_k &\to^* B'_k c_k \quad \text{w. prob. 1} \\
B'_k &\to^* a_{m+2} a_{m+3} ... a_{m+1+L'} \quad \text{w. prob. 1}
\end{aligned}
$$

in which $\to^*$ means that the left expands into the right through a sequence of rules that conform to the requirements of the Chomsky normal form (CNF, Definition 4.4). Hence the grammar $G_{m,L'}$ is in CNF.

The language of this grammar is

$$L(G_{m,L'}) = \{l_k = a_1 a_2 ... a_{m+1+L'} c_k : 1 \le k \le m\}.$$

The parse of an arbitrary l_k is shown in Figure 2. Each l_k corresponds to a unique parse determined by the choice of k. The structure of this PCFG is such that for the parsing algorithms we consider that proceed in a "left-to-right" fashion on l_k, before processing the last token c_k, it cannot infer the syntactic structure of $a_1 a_2 ... a_{m+1}$ any better than randomly guessing one of the m possibilities. This is the main intuition behind Theorems 2 and 5.

Remark. While our theorems focus on the limitation of "left-to-right" parsing, a symmetric argument implies the same limitation of "right-to-left" parsing. Thus, our claim is that *unidirectional* context (in either direction) limits the expressive power of parsing models.

3 Related Works

Neural models for parsing were first successfully implemented for supervised settings, e.g. (Vinyals et al., 2015; Dyer et al., 2016; Shen et al., 2018b). Unsupervised tasks remained seemingly out of reach, until the proposal of the Parsing-Reading-Predict Network (PRPN) by Shen et al. (2018a), whose performance was thoroughly verified by extensive experiments in (Htut et al., 2018). The

follow-up paper (Shen et al., 2019) introducing the ON-LSTM architecture simplified radically the architecture in (Shen et al., 2018a), while still ultimately attempting to fit a distance metric with the help of carefully designed master forget gates. Subsequent work by Kim et al. (2019a) departed from the usual way neural techniques are integrated in NLP, with great success: they proposed a neural parameterization for the EM algorithm for learning a PCFG, but in a manner that leverages semantic information as well — achieving a large improvement on unsupervised parsing tasks.[2]

In addition to constituency parsing, dependency parsing is another common task for syntactic parsing, but for our analyses on the ability of various approaches to represent the max-likelihood parse of sentences generated from PCFGs, we focus on the task of constituency parsing. Moreover, it's important to note that there is another line of work aiming to probe the ability of models trained without explicit syntactic consideration (e.g. BERT) to nevertheless discover some (rudimentary) syntactic elements (Bisk and Hockenmaier, 2015; Linzen et al., 2016; Choe and Charniak, 2016; Kuncoro et al., 2018; Williams et al., 2018; Goldberg, 2019; Htut et al., 2019; Hewitt and Manning, 2019; Reif et al., 2019). However, to-date, we haven't been able to extract parse trees achieving scores that are close to the oracle binarized trees on standard benchmarks (Kim et al., 2019b,a).

Methodologically, our work is closely related to a long line of works aiming to characterize the representational power of neural models (e.g. RNNs, LSTMs) through the lens of formal languages and formal models of computation. Some of the works of this flavor are empirical in nature (e.g. LSTMs have been shown to possess stronger abilities to recognize some context-free language and even some context-sensitive language, compared with simple RNNs (Gers and Schmidhuber, 2001; Suzgun et al., 2019) or GRUs (Weiss et al., 2018; Suzgun et al., 2019)); some results are theoretical in nature (e.g. Siegelmann and Sontag (1992)'s proof that with unbounded precision and unbounded time complexity, RNNs are Turing-complete; related results investigate RNNs with bounded precision and computation time (Weiss et al., 2018), as well as memory (Merrill, 2019; Hewitt et al., 2020). Our work contributes to this line of works, but focuses on the task of syntactic parsing instead.

[2]By virtue of not relying on bounded or unidirectional context, the Compound PCFG (Kim et al., 2019a) eschews the techniques in our paper. Specifically, by employing a bidirectional LSTM inference network in the process of constructing a tree given a sentence, the parsing is no longer "left-to-right".

4 Preliminaries

In this section, we define some basic concepts and introduce the architectures we will consider.

4.1 Probabilistic context-free grammar

First recall several definitions around formal language, especially probabilistic context free grammar:

Definition 4.1 (Probabilistic context-free grammar (PCFG))**.** Formally, a PCFG (Chomsky, 1956) is a 5-tuple $G = (\Sigma, N, S, R, \Pi)$ in which Σ is the set of terminals, N is the set of non-terminals, $S \in N$ is the start symbol, R is the set of production rules of the form $r = (r_L \to r_R)$, where $r_L \in N$, r_R is of the form $B_1 B_2 ... B_m$, $m \in \mathbb{Z}_+$, and $\forall i \in \{1, 2, ..., m\}, B_i \in (\Sigma \cup N)$. Finally, $\Pi : R \mapsto [0, 1]$ is the rule probability function, in which for any $r = (A \to B_1 B_2 ... B_m) \in R$,

$\Pi(r)$ is the conditional probability

$$P(r_R = B_1 B_2 ... B_m \,|\, r_L = A).$$

Definition 4.2 (Parse tree)**.** Let T_G denote the set of parse trees that G can derive. Each $t \in T_G$ is associated with $\texttt{yield(t)} \in \Sigma^*$, the sequence of terminals composed of the leaves of t and $P_T(t) \in [0, 1]$, the probability of the parse tree, defined by the product of the probabilities of the rules in the derivation of t.

Definition 4.3 (Language and sentence)**.** The language of G is

$$L(G) = \{s \in \Sigma^* : \exists t \in T_G, \texttt{yield(t)} = s\}.$$

Each $s \in L(G)$ is called a sentence in $L(G)$, and is associated with the set of parses $T_G(s) = \{t \in T_G \,|\, \texttt{yield(t)} = s\}$, the set of max likelihood parses, $\arg\max_{t \in T_G(s)} P_T(t)$, and its probability $P_S(s) = \sum_{t \in T_G(s)} P_T(t)$.

Definition 4.4 (Chomsky normal form (CNF))**.** A PCFG $G = (\Sigma, N, S, R, \Pi)$ is in CNF (Chomsky, 1959) if we require, in addition to Definition 4.1, that each rule $r \in R$ is in the form $A \to B_1 B_2$ where $B_1, B_2 \in N \setminus \{S\}$; $A \to a$ where $a \in \Sigma, a \neq \epsilon$; or $S \to \epsilon$ which is only allowed if the empty string $\epsilon \in L(G)$.

Every PCFG G can be converted into a PCFG G' in CNF such that $L(G) = L(G')$ (Hopcroft et al., 2006).

4.2 Syntactic distance

The Parsing-Reading-Predict Networks (PRPN) (Shen et al., 2018a) is one of the leading approaches to unsupervised constituency parsing. The parsing network (which computes the parse tree, hence the only part we focus on in our paper) is a convolutional network that computes the syntactic distances $d_t = d(w_{t-1}, w_t)$ (defined in Section 2.1) based on the past L words. A deterministic greedy tree induction algorithm is then used to produce a parse tree as follows. First, we split the sentence $w_1...w_n$ into two constituents, $w_1...w_{t-1}$ and $w_t...w_n$, where $t \in \text{argmax}\{d_t\}_{t=2}^n$ and form the left and right subtrees of t. We recursively repeat this procedure for the newly created constituents. An algorithmic form of this procedure is included as Algorithm 1 in Appendix A.

Note that, due to the deterministic nature of the tree-induction process, the ability of PRPN to learn a PCFG is completely contingent upon learning a good syntactic distance.

4.3 The ordered neuron architecture

Building upon the idea of representing the syntactic information with a real-valued distance measure at each position, a simple extension is to associate each position with a learned vector, and then use the vector for syntactic parsing. The ordered-neuron LSTM (ON-LSTM, Shen et al., 2019) proposes that the nodes that are closer to the root in the parse tree generate a longer span of terminals, and therefore should be less frequently "forgotten" than nodes that are farther away from the root. The difference in the frequency of forgetting is captured by a carefully designed master forget gate vector $\tilde{f}$, as shown in Figure 3 (in Appendix B). Formally:

Definition 4.5 (Master forget gates, Shen et al., 2019)**.** Given the input sentence $W = w_1 w_2 ... w_n$ and a trained ON-LSTM, running the ON-LSTM on W gives the master forget gates, which are a sequence of D-dimensional vectors $\{\tilde{f}_t\}_{t=1}^n$, in which at each position t, $\tilde{f}_t = \tilde{f}_t(w_1, ..., w_t) \in [0, 1]^D$. Moreover, let $\tilde{f}_{t,j}$ represent the j-th dimension of $\tilde{f}_t$. The ON-LSTM architectures requires that $\tilde{f}_{t,1} = 0$, $\tilde{f}_{t,D} = 1$, and

$$\forall i < j, \quad \tilde{f}_{t,i} \leq \tilde{f}_{t,j}.$$

When parsing a sentence, the real-valued master forget gate vector $\tilde{f}_t$ at each position t is reduced to a single real number representing the syntactic distance d_t at position t (see (1)) (Shen et al., 2018a). Then, use the syntactic distances to obtain a parse.

4.4 Transition-based parsing

In addition to outputting a single real numbered distance or a vector at each position t, a left-to-right model can also parse a sentence by outputting a sequence of "transitions" at each position t, an idea proposed in some traditional parsing approaches (Sagae and Lavie, 2005; Chelba, 1997; Chelba and Jelinek, 2000), and also some more recent neural parameterization (Dyer et al., 2016).

We introduce several items of notation:

- z_i^t: the i-th transition performed when reading in w_t, the t-th token of the sentence

$$W = w_1 w_2 ... w_n.$$

- N_t: the number of transitions performed between reading in the token w_t and reading in the next token w_{t+1}.
- Z_t: the sequence of transitions after reading in the prefix $w_1 w_2 ... w_t$ of the sentence.

$$Z_t = \{(z_1^j, z_2^j, ..., z_{N_j}^j) \,|\, j = 1..t\}.$$

- Z: the parse of the sentence W. $Z = Z_n$.

We base our analysis on the approach introduced in the *parsing* version of (Dyer et al., 2016), though that work additionally proposes a *generator* version. [3]

Definition 4.6 (Transition-based parser)**.** A *transition-based parser* uses a stack (initialized to empty) and an input buffer (initialized with the sentence $w_1...w_t$). At each position t, based on a context c_t, the parser outputs a sequence of parsing transitions $\{z_i^t\}_{i=1}^{N_t}$, where each z_i^t can be one of the following transitions (Definition 4.7). The parsing stops when the stack contains one single constituent, and the buffer is empty.

Definition 4.7 (Parser transitions, Dyer et al., 2016)**.** A parsing transition can be one of the following three types:

- NT(X) pushes a non-terminal X onto the stack.
- SHIFT: removes the first terminal from the input buffer and pushes onto the stack.

[3] Dyer et al. (2016) additionally proposes some *generator* transitions. For simplicity, we analyze the simplest form: we only allow the model to return one parse, composed of the *parser* transitions, for a given input sentence. Note that this simplified variant still confers full representational power in the "full context" setting (see Section 7).

- REDUCE: pops from the stack until an open non-terminal is encountered, then pops this non-terminal and assembles everything popped to form a new constituent, labels this new constituent using this non-terminal, and finally pushes this new constituent onto the stack.

In Appendix Section C, we provide an example of parsing the sentence "*I drink coffee with milk*" using the set of transitions given by Definition 4.7.

The different context specifications and the corresponding representational powers of the transition-based parser are discussed in Section 7.

5 Representational Power of Neural Syntactic Distance Methods

In this section we formalize the results on syntactic distance-based methods. Since the tree induction algorithm always generates a binary tree, we consider only PCFGs in Chomsky normal form (CNF) (Definition 4.4) so that the max likelihood parse of a sentence is also a binary tree structure.

To formalize the notion of "representing" a PCFG, we introduce the following definition:

Definition 5.1 (Representing PCFG with syntactic distance)**.** Let G be any PCFG in Chomsky Normal Form. A syntactic distance function d is said to be able to p**-represent** G if for a set of sentences in $L(G)$ whose total probability is at least p, d can correctly induce the tree structure of the max likelihood parse of these sentences without ambiguity.

Remark. Ambiguities could occur when, for example, there exists t such that $d_t = d_{t+1}$. In this case, the tree induction algorithm would have to break ties when determining the local structure for $w_{t-1}w_tw_{t+1}$. We preclude this possibility in Definition 5.1.

In the least restrictive setting, the whole sentence W, as well as the position index t can be taken into consideration when determining each d_t. We prove that under this setting, there is a syntactic distance measure that can represent any PCFG.

Theorem 1 (Full context)**.** *Let $c_t = (W, t)$. For each PCFG G in Chomsky normal form, there exists a syntactic distance measure $d_t = d(w_{t-1}, w_t \mid c_t)$ that can 1-represent G.*

Proof. For any sentence $s = s_1s_2...s_n \in L(G)$, let T be its max likelihood parse tree. Since G is in Chomsky normal form, T is a binary tree. We will describe an assignment of $\{d_t : 2 \leq t \leq n\}$ such that their order matches the level at which the branches split in T. Specifically, $\forall t \in [2, n]$, let a_t denote the lowest common ancestor of w_{t-1} and w_t in T. Let d'_t denote the shortest distance between a_t and the root of T. Finally, let $d_t = n - d'_t$. As a result, $\{d_t : 2 \leq t \leq n\}$ induces T. □

Remark. Since any PCFG can be converted to Chomsky normal form (Hopcroft et al., 2006), Theorem 1 implies that given the whole sentence and the position index as the context, the syntactic distance has sufficient representational power to capture any PCFG. It does not state, however, that the whole sentence and the position are the minimal contextual information needed for representability nor does it address training (i.e. optimization) issues.

On the flipside, we show that restricting the context even mildly can considerably decrease the representational power. Namely, we show that if context is bounded *even in a single direction* (to the left or to the right), there are PCFGs on which any syntactic distance will perform poorly [4]. (Note in the implementation (Shen et al., 2018a) the context only considers a bounded window to the left.)

Theorem 2 (Limitation of left-to-right parsing via syntactic distance)**.** *Let $w_0 = \langle S \rangle$ be the sentence start symbol. Let the context*

$$c_t = (w_0, w_1, ..., w_{t+L'}).$$

$\forall \epsilon > 0$, there exists a PCFG G in Chomsky normal form, such that any syntactic distance measure $d_t = d(w_{t-1}, w_t \mid c_t)$ cannot ϵ-represent G.

Proof. Let $m > 1/\epsilon$ be a positive integer. Consider the PCFG $G_{m,L'}$ in Definition 2.1.

For any $k \in [m]$, consider the string $l_k \in L(G_{m,L'})$. Note that in the parse tree of l_k, the rule $S \to A_kB_k$ is applied. Hence, a_k and a_{k+1} are the unique pair of adjacent non-terminals in $a_1a_2...a_{m+1}$ whose lowest common ancestor is the closest to the root in the parse tree of l_k. Then, in order for the syntactic distance metric d to induce the correct parse tree for l_k, d_k must be the unique maximum in $\{d_t : 2 \leq t \leq m + 1\}$.

However, d is restricted to be in the form

$$d_t = d(w_{t-1}, w_t \mid w_0, w_1, ..., w_{t+L'}).$$

[4] In Theorem 2 we prove the more typical case, i.e. unbounded left context and bounded right context. The other case, i.e. bounded left context and unbounded right context, can be proved symmetrically.

Note that $\forall 1 \leq k_1 < k_2 \leq m$, the first $m+1+L'$ tokens of l_{k_1} and l_{k_2} are the same, which implies that the inferred syntactic distances

$$\{d_t : 2 \leq t \leq m+1\}$$

are the same for l_{k_1} and l_{k_2} at each position t. Thus, it is impossible for d to induce the correct parse tree for both l_{k_1} and l_{k_2}. Hence, d is correct on at most one $l_k \in L(G_{m,L'})$, which corresponds to probability at most $1/m < \epsilon$. Therefore, d cannot ϵ-represent $G_{m,L'}$. □

Remark. In the counterexample, there are only m possible parse structures for the prefix $a_1 a_2 ... a_{m+1}$. Hence, the proved fact that the probability of being correct is at most $1/m$ means that under the restrictions of unbounded look-back and bounded look-ahead, the distance cannot do better than random guessing for this grammar.

Remark. The above Theorem 2 formalizes the intuition discussed in (Htut et al., 2018) outlining an intrinsic limitation of only considering bounded context in one direction. Indeed, for the PCFG constructed in the proof, the failure is a function of the context, not because of the fact that we are using a distance-based parser.

Note that as a corollary of the above theorem, if there is no context ($c_t = \texttt{null}$) or the context is both bounded and unidirectional, i.e.

$$c_t = w_{t-L} w_{t-L+1} ... w_{t-1} w_t,$$

then there is a PCFG that cannot be ϵ-represented by any such d.

6 Representational Power of the Ordered Neuron Architecture

In this section, we formalize the results characterizing the representational power of the ON-LSTM architecture. The master forget gates of the ON-LSTM, $\{\tilde{f}_t\}_{t=2}^n$ in which each $\tilde{f}_t \in [0,1]^D$, encode the hierarchical structure of a parse tree, and Shen et al. (2019) proposes to carry out unsupervised constituency parsing via a reduction from the gate vectors to syntactic distances by setting:

$$\hat{d}_t^f = D - \sum_{j=1}^{D} \tilde{f}_{t,j} \text{ for } t = 2..n \qquad (1)$$

First we show that the gates in ON-LSTM *in principle* form a lossless representation of any parse tree.

Theorem 3 (Lossless representation of a parse tree)**.** *For any sentence $W = w_1 w_2 ... w_n$ with parse tree T in any PCFG in Chomsky normal form, there exists a dimensionality $D \in \mathbb{Z}_+$, a sequence of vectors $\{\tilde{f}_t\}_{t=2}^n$ in which each $\tilde{f}_t \in [0,1]^D$, such that the estimated syntactic distances via (1) induce the structure of T.*

Proof. By Theorem 1, there is a syntactic distance measure $\{d_t\}_{t=2}^n$ that induces the structure of T (such that $\forall t, d_t \neq d_{t+1}$).

For each $t = 2..n$, set $\hat{d}_t = k$ if d_t is the k-th smallest entry in $\{d_t\}_{t=2}^n$, breaking ties arbitrarily. Then, each $\hat{d}_t \in [1, n-1]$, and $\{\hat{d}_t\}_{t=2}^n$ also induces the structure of T.

Let $D = n-1$. For each $t = 2..n$, let $\tilde{f}_t = (0, ..., 0, 1, ..., 1)$ whose lower $\hat{d}_t$ dimensions are 0 and higher $D - \hat{d}_t$ dimensions are 1. Then,

$$\hat{d}_t^f = D - \sum_{j=1}^{D} \tilde{f}_{t,j} = D - (D - \hat{d}_t) = \hat{d}_t.$$

Therefore, the calculated $\{\hat{d}_t^f\}_{t=2}^n$ induces the structure of T. □

Although Theorem 3 shows the ability of the master forget gates to perfectly represent any parse tree, a *left-to-right* parsing can be proved to be unable to return the correct parse with high probability. In the actual implementation in (Shen et al., 2019), the (real-valued) master forget gate vectors $\{\tilde{f}_t\}_{t=1}^n$ are produced by feeding the input sentence $W = w_1 w_2 ... w_n$ to a model trained with a language modeling objective. In other words, $\tilde{f}_{t,j}$ is calculated as a function of $w_1, ..., w_t$, rather than the entire sentence.

As such, this left-to-right parser is subject to similar limitations as in Theorem 2:

Theorem 4 (Limitation of syntactic distance estimation based on ON-LSTM)**.** *For any $\epsilon > 0$, there exists a PCFG G in Chomsky normal form, such that the syntactic distance measure calculated with (1), $\hat{d}_t^f$, cannot ϵ-represent G.*

Proof. Since by Definition 4.5, $\tilde{f}_{t,j}$ is a function of $w_1, ..., w_t$, the estimated syntactic distance $\hat{d}_t^f$ is also a function of $w_1, ..., w_t$. By Theorem 2, even with unbounded look-back context $w_1, ..., w_t$, there exists a PCFG for which the probability that $\hat{d}_t^f$ induces the correct parse is arbitrarily low. □

7 Representational Power of Transition-Based Parsing

In this section, we analyze a transition-based parsing framework inspired by (Dyer et al., 2016; Chelba and Jelinek, 2000; Chelba, 1997).

Again, we proceed to say first that "full context" confers full representational power. Namely, using the terminology of Definition 4.6, we let the context c_t at each position t be the whole sentence W and the position index t. Note that any parse tree can be generated by a sequence of transitions defined in Definition 4.7. Indeed, Dyer et al. (2016) describes an algorithm to find such a sequence of transitions via a "depth-first, left-to-right traversal" of the tree.

Proceeding to limited context, in the setting of typical left-to-right parsing, the context c_t consists of all current and past tokens $\{w_j\}_{j=1}^{t}$ and all previous parses $\{(z_1^j, ..., z_{N_j}^j)\}_{j=1}^{t}$. We'll again prove even stronger negative results, where we allow an optional look-ahead to L' input tokens to the right.

Theorem 5 (Limitation of transition-based parsing without full context)**.** *For any $\epsilon > 0$, there exists a PCFG G in Chomsky normal form, such that for any learned transition-based parser (Definition 4.6) based on context*

$$c_t = (\{w_j\}_{j=1}^{t+L'}, \{(z_1^j, ..., z_{N_j}^j)\}_{j=1}^{t}),$$

the sum of the probabilities of the sentences in $L(G)$ for which the parser returns the maximum likelihood parse is less than ϵ.

Proof. Let $m > 1/\epsilon$ be a positive integer. Consider the PCFG $G_{m,L'}$ in Definition 2.1.

Note that $\forall k, S \rightarrow A_k B_k$ is applied to yield string l_k. Then in the parse tree of l_k, a_k and a_{k+1} are the unique pair of adjacent terminals in $a_1 a_2 ... a_{m+1}$ whose lowest common ancestor is the closest to the root. Thus, different l_k requires a different sequence of transitions within the first $m+1$ input tokens, i.e. $\{z_i^t\}_{i \geq 1, 1 \leq t \leq m+1}$.

For each $w \in L(G_{m,L'})$, before the last token $w_{m+2+L'}$ is processed, based on the common prefix $w_1 w_2 ... w_{m+1+L'} = a_1 a_2 ... a_{m+1+L'}$, it is equally likely that $w = l_k, \forall k$, w. prob. $1/m$ each.

Moreover, when processing w_{m+1}, the bounded look-ahead window of size L' does not allow access to the final input token $a_{m+2+L'} = c_k$.

Thus, $\forall 1 \leq k_1 < k_2 \leq m$, it is impossible for the parser to return the correct parse tree for both l_{k_1} and l_{k_2} without ambiguity. Hence, the parse is correct on at most one $l_k \in L(G)$, which corresponds to probability at most $1/m < \epsilon$. □

8 Conclusion

In this work, we considered the representational power of two frameworks for constituency parsing prominent in the literature, based on learning a syntactic distance and learning a sequence of iterative transitions to build the parse tree — in the sandbox of PCFGs. In particular, we show that if the context for calculating distance/deciding on transitions is limited at least to one side (which is typically the case in practice for existing architectures), there are PCFGs for which no good distance metric/sequence of transitions can be chosen to construct the maximum likelihood parse.

This limitation was already suspected in (Htut et al., 2018) as a potential failure mode of leading neural approaches like (Shen et al., 2018a, 2019) and we show formally that this is the case. The PCFGs with this property track the intuition that bounded context methods will have issues when the parse at a certain position depends heavily on latter parts of the sentence.

The conclusions thus suggest re-focusing our attention on methods like (Kim et al., 2019a) which have enjoyed greater success on tasks like unsupervised constituency parsing, and do not fall in the paradigm analyzed in our paper. A question of definite further interest is how to augment models that have been successfully scaled up (e.g. BERT) in a principled manner with syntactic information, such that they can capture syntactic structure (like PCFGs). The other question of immediate importance is to understand the interaction between the syntactic and semantic modules in neural architectures — information is shared between such modules in various successful architectures, e.g. (Dyer et al., 2016; Shen et al., 2018a, 2019; Kim et al., 2019a), and the relative pros and cons of doing this are not well understood. Finally, our paper purely focuses on representational power, and does not consider algorithmic and statistical aspects of training. As any model architecture is associated with its distinct optimization and generalization considerations, and natural language data necessitates the modeling of the interaction between syntax and semantics, those aspects of considerations are well beyond the scope of our analysis in this paper using the controlled sandbox of PCFGs, and are interesting directions for future work.

References

Yonatan Bisk and Julia Hockenmaier. 2015. Probing the linguistic strengths and limitations of unsupervised grammar induction. 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 1395–1404, Beijing, China. Association for Computational Linguistics.

Ciprian Chelba. 1997. A structured language model. In *35th Annual Meeting of the Association for Computational Linguistics and 8th Conference of the European Chapter of the Association for Computational Linguistics*, pages 498–500, Madrid, Spain. Association for Computational Linguistics.

Ciprian Chelba and Frederick Jelinek. 2000. Structured language modeling. *Computer Speech & Language*, 14(4):283 – 332.

Qian Chen, Xiaodan Zhu, Zhen-Hua Ling, Si Wei, Hui Jiang, and Diana Inkpen. 2017. Enhanced LSTM for natural language inference. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1657–1668, Vancouver, Canada. Association for Computational Linguistics.

Do Kook Choe and Eugene Charniak. 2016. Parsing as language modeling. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2331–2336, Austin, Texas. Association for Computational Linguistics.

N. Chomsky. 1956. Three models for the description of language. *IRE Transactions on Information Theory*, 2(3):113–124.

Noam Chomsky. 1959. On certain formal properties of grammars. *Information and Control*, 2(2):137 – 167.

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. Association for Computational Linguistics.

Chris Dyer, Adhiguna Kuncoro, Miguel Ballesteros, and Noah A. Smith. 2016. Recurrent neural network grammars. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 199–209, San Diego, California. Association for Computational Linguistics.

F. Gers and J. Schmidhuber. 2001. Lstm recurrent networks learn simple context-free and context-sensitive languages. *IEEE transactions on neural networks*, 12 6:1333–40.

Yoav Goldberg. 2019. Assessing bert's syntactic abilities.

Qi He, Han Wang, and Yue Zhang. 2020. Enhancing generalization in natural language inference by syntax. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 4973–4978, Online. Association for Computational Linguistics.

John Hewitt, Michael Hahn, Surya Ganguli, Percy Liang, and Christopher D. Manning. 2020. RNNs can generate bounded hierarchical languages with optimal memory. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1978–2010, Online. Association for Computational Linguistics.

John Hewitt and Christopher D. Manning. 2019. A structural probe for finding syntax in word representations. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4129–4138, Minneapolis, Minnesota. Association for Computational Linguistics.

John E. Hopcroft, Rajeev Motwani, and Jeffrey D. Ullman. 2006. *Introduction to Automata Theory, Languages, and Computation (3rd Edition)*. Addison-Wesley Longman Publishing Co., Inc., USA.

Phu Mon Htut, Kyunghyun Cho, and Samuel Bowman. 2018. Grammar induction with neural language models: An unusual replication. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 371–373, Brussels, Belgium. Association for Computational Linguistics.

Phu Mon Htut, Jason Phang, Shikha Bordia, and Samuel R. Bowman. 2019. Do attention heads in bert track syntactic dependencies? *ArXiv*, abs/1911.12246.

Yoon Kim, Chris Dyer, and Alexander Rush. 2019a. Compound probabilistic context-free grammars for grammar induction. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2369–2385, Florence, Italy. Association for Computational Linguistics.

Yoon Kim, Alexander Rush, Lei Yu, Adhiguna Kuncoro, Chris Dyer, and Gábor Melis. 2019b. Unsupervised recurrent neural network grammars. 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 1105–1117, Minneapolis, Minnesota. Association for Computational Linguistics.

Adhiguna Kuncoro, Chris Dyer, John Hale, Dani Yogatama, Stephen Clark, and Phil Blunsom. 2018. LSTMs can learn syntax-sensitive dependencies well, but modeling structure makes them better. In

Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 1426–1436, Melbourne, Australia. Association for Computational Linguistics.

Tal Linzen, Emmanuel Dupoux, and Yoav Goldberg. 2016. Assessing the ability of LSTMs to learn syntax-sensitive dependencies. *Transactions of the Association for Computational Linguistics*, 4:521–535.

William Merrill. 2019. Sequential neural networks as automata. In *Proceedings of the Workshop on Deep Learning and Formal Languages: Building Bridges*, pages 1–13, Florence. Association for Computational Linguistics.

Deric Pang, Lucy H. Lin, and Noah A. Smith. 2019. Improving natural language inference with a pretrained parser.

Emily Reif, Ann Yuan, Martin Wattenberg, Fernanda B Viegas, Andy Coenen, Adam Pearce, and Been Kim. 2019. Visualizing and measuring the geometry of bert. In *Advances in Neural Information Processing Systems*, volume 32, pages 8594–8603. Curran Associates, Inc.

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

Yikang Shen, Zhouhan Lin, Chin wei Huang, and Aaron Courville. 2018a. Neural language modeling by jointly learning syntax and lexicon. In *International Conference on Learning Representations*.

Yikang Shen, Zhouhan Lin, Athul Paul Jacob, Alessandro Sordoni, Aaron Courville, and Yoshua Bengio. 2018b. Straight to the tree: Constituency parsing with neural syntactic distance. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1171–1180, Melbourne, Australia. Association for Computational Linguistics.

Yikang Shen, Shawn Tan, Alessandro Sordoni, and Aaron Courville. 2019. Ordered neurons: Integrating tree structures into recurrent neural networks. In *International Conference on Learning Representations*.

Hava T. Siegelmann and Eduardo D. Sontag. 1992. On the computational power of neural nets. In *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, COLT '92, page 440–449, New York, NY, USA. Association for Computing Machinery.

Mirac Suzgun, Yonatan Belinkov, Stuart Shieber, and Sebastian Gehrmann. 2019. LSTM networks can perform dynamic counting. In *Proceedings of the Workshop on Deep Learning and Formal Languages: Building Bridges*, pages 44–54, Florence. Association for Computational Linguistics.

Oriol Vinyals, Ł ukasz Kaiser, Terry Koo, Slav Petrov, Ilya Sutskever, and Geoffrey Hinton. 2015. Grammar as a foreign language. In *Advances in Neural Information Processing Systems*, volume 28, pages 2773–2781. Curran Associates, Inc.

Gail Weiss, Yoav Goldberg, and Eran Yahav. 2018. On the practical computational power of finite precision RNNs for language recognition. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 740–745, Melbourne, Australia. Association for Computational Linguistics.

Adina Williams, Andrew Drozdov, and Samuel R. Bowman. 2018. Do latent tree learning models identify meaningful structure in sentences? *Transactions of the Association for Computational Linguistics*, 6:253–267.

A Tree Induction Algorithm Based on Syntactic Distance

The following algorithm is proposed in (Shen et al., 2018a) to create a parse tree based on a given syntactic distance.

Algorithm 1: Tree induction based on syntactic distance

Data: Sentence $W = w_1 w_2 ... w_n$, syntactic distances $d_t = d(w_{t-1}, w_t \mid c_t)$, $2 \leq t \leq n$
Result: A parse tree for W
Initialize the parse tree with a single node $n_0 = w_1 w_2 ... w_n$;
while $\exists$ *leaf node* $n = w_i w_{i+1} ... w_j$ *where* $i < j$ **do**
  Find $k \in \arg\max_{i+1 \leq k \leq j} d_k$;
  Create the left child n_l and the right child n_r of n ;
  $n_l \leftarrow w_i w_{i+1} ... w_{k-1}$;
  $n_r \leftarrow w_k w_{k+1} ... w_j$;
end
return The parse tree rooted at n_0.

B ON-LSTM Intuition

See Figure 3 below, which is excerpted from (Shen et al., 2019) with minor adaptation to the notation.

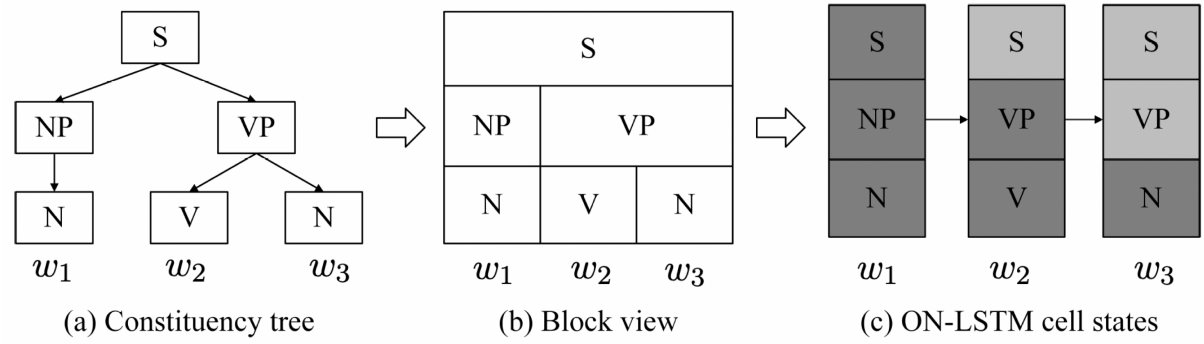
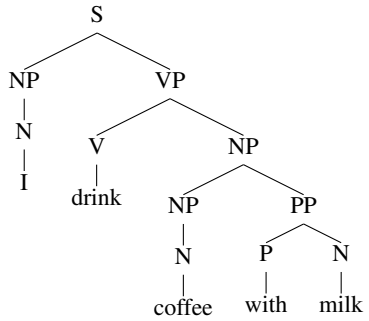


Figure 3: Relationship between the parse tree, the block view, and the ON-LSTM. Excerpted from (Shen et al., 2019) with minor adaptation to the notation.

C Examples of parsing transitions

Table 1 below shows an example of parsing the sentence "*I drink coffee with milk*" using the set of transitions given by Definition 4.7, which employs the parsing framework of (Dyer et al., 2016). The parse tree of the sentence is given by



Stack	**Buffer**	**Action**
	I drink coffee with milk	`NT(S)`
(S	*I drink coffee with milk*	`NT(NP)`
(S | (NP	*I drink coffee with milk*	`NT(N)`
(S | (NP | (N	*I drink coffee with milk*	`SHIFT`
(S | (NP | (N | *I*	*drink coffee with milk*	`REDUCE`
(S | (NP (N *I*))	*drink coffee with milk*	`NT(VP)`
(S | (NP (N *I*)) | (VP	*drink coffee with milk*	`NT(V)`
(S | (NP (N *I*)) | (VP | (V	*drink coffee with milk*	`SHIFT`
(S | (NP (N *I*)) | (VP | (V | *drink*	*coffee with milk*	`REDUCE`
(S | (NP (N *I*)) | (VP | (V *drink*)	*coffee with milk*	`NT(NP)`
(S | (NP (N *I*)) | (VP | (V | *drink*) | (NP	*coffee with milk*	`NT(NP)`
(S | (NP (N *I*)) | (VP | (V *drink*) | (NP | (NP	*coffee with milk*	`NT(N)`
(S | (NP (N *I*)) | (VP | (V *drink*) | (NP | (NP | (N	*coffee with milk*	`SHIFT`
(S | (NP (N *I*)) | (VP | (V *drink*) | (NP | (NP | (N | *coffee*	*with milk*	`REDUCE`
(S | (NP (N *I*)) | (VP | (V *drink*) | (NP | (NP (N *coffee*))	*with milk*	`NT(PP)`
(S | (NP (N *I*)) | (VP | (V *drink*) | (NP | (NP (N *coffee*)) | (PP	*with milk*	`NT(P)`
(S | (NP (N *I*)) | (VP | (V *drink*) | (NP | (NP (N *coffee*)) | (PP | (P	*with milk*	`SHIFT`
(S | (NP (N *I*)) | (VP | (V *drink*) | (NP | (NP (N *coffee*)) | (PP | (P | *with*	*milk*	`REDUCE`
(S | (NP (N *I*)) | (VP | (V *drink*) | (NP | (NP (N *coffee*)) | (PP | (P *with*)	*milk*	`NT(N)`
(S | (NP (N *I*)) | (VP | (V *drink*) | (NP | (NP (N *coffee*)) | (PP | (P *with*) | (N	*milk*	`SHIFT`
(S | (NP (N *I*)) | (VP | (V *drink*) | (NP | (NP (N *coffee*)) | (PP | (P *with*) | (N | *milk*		`REDUCE`
(S (NP (N *I*)) (VP (V *drink*) (NP (NP (N *coffee*)) (PP (P *with*) (N *milk*)))))		

Table 1: Transition-based parsing of the sentence "*I drink coffee with milk*".