Is Incremental Structure Prediction Process Universal across Languages?: Revisiting Parsing Strategy through Speculation

Anonymous ACL submission

Abstract

While natural language is processed incrementally, it is unclear whether the syntactic structure prediction process is universal across languages or language-specific. This study investigates this question by revisiting parsing strategies of syntactic language models that incrementally predict both the next token and the associated syntactic structure. Unlike previous studies that have focused on a few strategies, we examine a wide range of strategies by introducing different parameterizations of "speculation", which quantifies the degree to which a model predicts syntactic structure before encountering the corresponding tokens. The experiments with 10 typologically diverse languages reveal that the optimal strategy differs depending on the language and the beam size.

1 Introduction

Understanding how syntactic structure is incrementally processed during language comprehension is a fundamental challenge in computational linguistics and cognitive science. Syntactic language modeling (SLM), also known as syntax-aware language modeling, provides a direct approach to addressing this question (Choe and Charniak, 2016; Dyer et al., 2016; Qian et al., 2021; Sartran et al., 2022). SLM is a task that jointly performs parsing and next-token prediction, thereby explicitly modeling the interplay between syntactic structure and incremental sequence processing. This approach has proven valuable for offering insights into the cognitive mechanisms of human language processing (Hale et al., 2018; Yoshida et al., 2021; Sugimoto et al., 2024).

While SLM provides a framework for modeling syntactic processing, there exist multiple ways to incrementally process the same sequence of tokens and syntactic structures depending on the timing of structure prediction (Figure 1). These differences in processing are captured by the concept of "parsing strategy," (Abney and Johnson, 1991). For example, Figure 1 illustrates the two most commonly used strategies in parsing: top-down and bottom-up. Top-down is a strategy that predicts structure before tokens, while bottom-up is a strategy that predicts structure after tokens. Previous studies in SLM, however, have primarily focused on a limited set of strategies, such as top-down, bottom-up, and left-corner, and, moreover, lack cross-linguistic comparisons (Kuncoro et al., 2018; Yoshida et al., 2021), leaving it unclear whether optimal strategies are universal or language-specific.

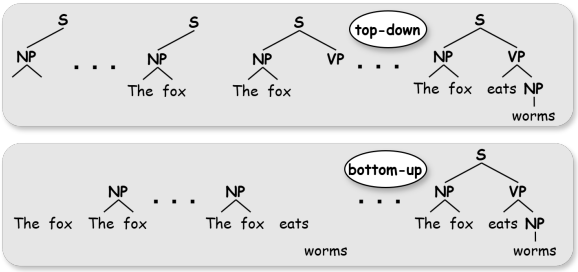

Figure 1: Example of incremental structure prediction process.

This paper aims to address this gap in the literature by conducting a comprehensive analysis of parsing strategies for SLM across a diverse set of languages. To this end, we explore a wide range of parsing strategies from the perspective of "speculation", which quantifies the degree to which a model predicts syntactic structure before encountering the corresponding tokens. For example, the top-down strategy is highly speculative because it cannot use token information for structure prediction, and the predicted structure may be incorrect depending on subsequent tokens. In this work, we consider strategies based on 4 different parameterizations of speculation and evaluate a total of 15 distinct strategies on SLM tasks in 10 typologically diverse languages. While less speculative strategies might intuitively seem more advantageous, our ex-

periment demonstrates that it is not always the case: the optimal strategy can vary across languages and depends on the beam size. Furthermore, we also analyze the fundamental question: does syntactic structure contribute to token prediction? By comparing strategies with different degrees of speculation, we show that syntactic structure indeed captures information about tokens, while also suggesting that exact parsing might not be necessary for token prediction.

2 Background

Early studies argued that the left-corner strategy is more efficient and cognitively plausible than top-down or bottom-up strategies (Abney and Johnson, 1991; Resnik, 1992).[1] These arguments relied primarily on analyzing the maximum stack size required by shift-reduce parsers (Abney and Johnson, 1991; Resnik, 1992; Noji and Miyao, 2014). However, as Resnik (1992) points out, the difference in stack efficiency between strategies depends on the specific implementation of the parser. For instance, implementations of Recurrent Neural Network Grammar (RNNG) (Dyer et al., 2016; Noji and Oseki, 2021) require $O(n)$ stack size for right-branching structures even with the top-down strategy unlike claimed to be $O(1)$ in (Abney and Johnson, 1991). Therefore, it is unclear to what extent stack efficiency influences the choice of incremental processing strategies.

In the context of SLM, recent studies have explored the impact of different parsing strategies on downstream tasks such as language modeling and parsing. For example, Kuncoro et al. (2018) compared top-down, bottom-up, and left-corner strategies for English number agreement, finding that top-down parsing yielded better performance. Yoshida et al. (2021) compared top-down and left-corner strategies for Japanese language modeling, demonstrating the effectiveness of the left-corner strategy. Kuribayashi et al. (2024) compared top-down and left-corner strategies using an artificial language dataset with varying word order. However, these studies are limited in several aspects. First, they focus on a limited set of parsing strategies, e.g., top-down, bottom-up, and left-corner, due to the ease of implementation. Second, there is a lack of comprehensive cross-linguistic comparisons using real-world natural language data, leaving it unclear whether optimal parsing strategies are universal or language-specific.

To this end, this study conducts a more comprehensive analysis of parsing strategies for SLM, both in terms of strategies and languages.

[1]The left-corner strategy predicts a phrase structure immediately after reading the leftmost token of that phrase.

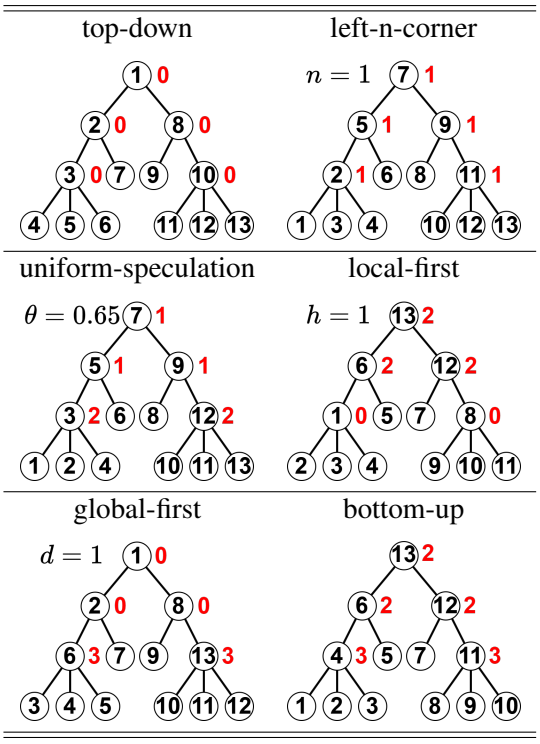

Table 1: Examples of parsing strategies. The numbers inside the circles indicate the order of node enumeration, and the numbers to the right of each nonterminal node represent its i_v.

3 Formulating Various Strategies

Following the general formulation of Abney and Johnson (1991), we formalize various parsing strategies. The difference between parsing strategies is defined by the timing at which each nonterminal node is opened. This allows us to express each strategy as a specific enumeration order of the nodes in a syntactic tree. Abney and Johnson (1991) demonstrated that different parsing strategies can be represented by strategy parameters i_v for each node v.[2] Let $u_1, \ldots, u_n$ be the children of v; $i_v = i$ indicates that the parent node v is opened immediately after its i-th child u_i is completed. The case of $i_v = 0$ indicates that v is opened before any of its children are created. By assigning i_v to every node v in a given syntactic tree, we can uniquely determine an incremental process of

[2]While Abney and Johnson (1991) originally defined the parameters for grammar rules, we generalize it to the nodes in syntactic trees.

predicting the syntactic tree. Strategies represented by this parameterization are called syntax-directed strategies (Abney and Johnson, 1991).

In this study, we formulate a variety of distinct strategies within the class of syntax-directed strategies to investigate whether the optimal strategy is language-universal or language-specific. Our formulation is based on the concept of "speculation", which refers to the degree to which a model predicts syntactic structure before encountering the corresponding tokens. We consider 4 different parameterizations of speculation, each capturing a different aspect of this concept. By exploring multiple parameter settings within each parameterization, we analyze a total of 15 strategies. Table 1 shows some examples of the strategies used in this study. Note that both top-down and bottom-up strategies can be expressed by specific parameter settings within any of the four parameterizations.

3.1 Left-n-corner strategy

Besides top-down and bottom-up strategies, the left-corner strategy is another major strategy used in parsing research. In this study, we also experiment with a generalization of the left-corner strategy formulated by Abney and Johnson (1991), which we refer to as the "left-n-corner strategy".[3] In a left-n-corner strategy, the parent node v is predicted after at most n of its children have been completed. Formally, left-n-corner strategies are defined by a speculation parameter n as $i_v = \min(n, n_v)$. When $n = 0$, the left-n-corner strategy is equivalent to the top-down strategy. When $n = \infty$, it is equivalent to the bottom-up strategy.

3.2 Uniform-speculation Strategy

In the left-n-corner strategies, the number of children completed before predicting the parent n is constant for all nodes. However, with this parameterization, whether the timing of opening the parent node v is closer to top-down or bottom-up can vary across nodes, depending on the number of children n_v. Therefore, in this study, we introduce strategies in which the timing of opening the parent node v is less dependent on n_v and is consistent across all nodes.

Intuitively, this strategy, which we call the "uniform-speculation strategy", is defined by a real-valued speculation parameter $\theta \in [0, 1]$, representing the proportion of children created before the parent. For a node v with n_v children, i_v is calculated as $i_v = \lfloor \theta \cdot (n_v + 1) \rfloor$. Here, $\theta \to 0$ corresponds to strategies closer to top-down, while $\theta \to 1$ corresponds to strategies closer to bottom-up.

3.3 Local/global-first Strategy

The two strategies discussed above, left-n-corner and uniform-speculation, determine the timing of opening a node v independently of its position within the syntactic tree. In this study, we also analyze strategies where the timing of opening v – that is, the degree of speculation – varies depending on whether v belongs to a local or global structure.

Defining whether a structure is local or global is not trivial. Here, we use the height and depth of each node to define local and global structures, and use these as parameters to control the degree of speculation of the strategies. Intuitively, nodes closer to leaf nodes, i.e., nodes with smaller height, are considered local, while nodes closer to the root node, i.e., nodes with smaller depth, are considered global.

First, we consider a "local-first strategy", which predicts local structures in a top-down manner and global structures in a bottom-up manner. Specifically, the speculation parameter of this strategy is a height threshold h:

$$i_v = \begin{cases} 0, & \text{if } h_v \leq h \\ n_v, & \text{otherwise} \end{cases}$$

where h_v is the height of node v.[4]

Similarly, we can also consider a "global-first strategy", which predicts global structures in a top-down manner and local structures in a bottom-up manner. This strategy is parameterized by a depth threshold d as follows:

$$i_v = \begin{cases} 0, & \text{if } d_v \leq d \\ n_v, & \text{otherwise} \end{cases}$$

where d_v is the depth of node v.[5]

When $h \to \infty$, the local-first strategy is closer to top-down, and when $h = 0$, it is equivalent to bottom-up. Similarly, when $d \to \infty$, the global-first strategy is closer to top-down, and when $d < 0$, it becomes bottom-up.

[3] This formulation is called "uniform syntax-directed strategy" in (Abney and Johnson, 1991). However, we use the name left-n-corner instead to emphasize that it is a generalization of the left-corner strategy.

[4] We define the height of leaf nodes to be 0.

[5] We define the depth of the root node to be 0.

4 Shift-reduce Syntactic Language Modeling

This section formalizes the syntactic language modeling task (SLM). In SLM, structure prediction is typically performed by a shift-reduce parser with a stack (Dyer et al., 2016; Noji and Oseki, 2021; Choe and Charniak, 2016; Qian et al., 2021; Sartran et al., 2022; Kuncoro et al., 2018). Stack-based parsing is performed by predicting a sequence of actions defined as stack operations. However, previous work designed a separate action set for each parsing strategy, making it difficult to handle various strategies within a unified framework (Kuncoro et al., 2018). To address this limitation, we generalize the action set used by a shift-reduce parser to represent a wide range of strategies with a single, unified set of actions.

4.1 Generalizing Shift-reduce Actions

A simple approach to represent various strategies with a single action set is to extend the stack operations beyond push and pop to include an "insert" operation. This allows us to open nonterminal nodes at different positions within the stack, effectively controlling the timing of structure prediction. Specifically, we define the following action set:

- NT(X; n): Inserts an open nonterminal node "(X" at the n-th position from the top of the stack, opening a phrase with category X. Note that a new phrase cannot be opened deeper than any already open phrase.[6]
- SHIFT: Pushes the next token onto the stack.
- REDUCE: Completes the topmost open phrase on the stack, popping and combining its elements into a single constituent.

While strategies other than top-down typically require a special FINISH action to terminate the parsing process (Kuncoro et al., 2018), we do not explicitly introduce a FINISH action. Instead, we terminate the parsing process when the end-of-sentence (EOS) token is shifted. This simplifies the formulation of syntactic language modeling and the beam search procedure, which will be described later.

This generalized action set can represent various parsing strategies by restricting how actions are selected. For example, if the position to open a phrase is always $n = 0$, i.e., the top of the stack, the strategy becomes equivalent to top-down. If REDUCE action is always performed immediately after NT(X; n) action, the strategy becomes equivalent to bottom-up, because the prediction of a phrase with n children always occurs after all its children are completed.

[6] This restriction is for implementation simplicity.

4.2 Model Formulation

First, we introduce the notations used to formulate SLM. Let $\mathcal{A}$ be the set of actions defined above. We define $A_k \subset \mathcal{A}^*$ as the set of action sequences that contain exactly k SHIFT actions and end with a SHIFT action. For an action sequence $a = (a_1, \ldots, a_T)$, let l_i denote the index of the i-th SHIFT action a_{l_i} in a.

Given a token sequence x and an action sequence a, the syntactic language model $\mathcal{M}$ defines the following joint probability:

$$p^{\mathcal{M}}_{\text{joint}}(x, a) \equiv \prod_{t=1}^{|a|} p^{\mathcal{M}}_{\text{action}}(a_t \mid a_{<t}, x_{\leq s(a_{<t})}) \tag{293}$$

$$\cdot \prod_{i=1}^{|x|} p^{\mathcal{M}}_{\text{token}}(x_i \mid a_{<l_i}, x_{<i}), \tag{294}$$

where $p^{\mathcal{M}}_{\text{joint}}$ is the joint distribution of the token sequence and the parsing action sequence, $p^{\mathcal{M}}_{\text{action}}$ is the conditional probability of the next parsing action, $p^{\mathcal{M}}_{\text{token}}$ is the conditional probability of the next token, and $s(a_{<t})$ denotes the number of SHIFT actions in the given action sequence. While the probability of generating a token is not typically separated into $p^{\mathcal{M}}_{\text{action}}$ and $p^{\mathcal{M}}_{\text{token}}$ in the formulation, the probabilities are typically separated in the implementations (Dyer et al., 2016; Noji and Oseki, 2021). Here, we introduce a formulation that aligns more closely with actual implementations. During supervised training, the model is trained to maximize $\log p^{\mathcal{M}}_{\text{joint}}(x, a)$ on the train dataset.

The probability distribution over token sequences of length $|x|$ is computed as follows:

$$p^{\mathcal{M}}(x) = \sum_{a \in A_{|x|}} p^{\mathcal{M}}_{\text{joint}}(x, a). \tag{311}$$

To calculate the probability distribution over sentences of arbitrary length, one can simply calculate $p^{\mathcal{M}}$ for token sequences x that end with the EOS token.

4.3 Modeling Incremental Inference Process

The goal of this study is to evaluate the incremental structure prediction process in natural language.

Previous work on SLM has primarily focused on evaluating models by approximating $p^{\mathcal{M}}$ using a trained model $\mathcal{M}$.

Approaches to approximating $p^{\mathcal{M}}$ in SLM can be broadly categorized into two types. The first approach uses candidate actions $\tilde{A}$ obtained from an external parser (Dyer et al., 2016; Kuncoro et al., 2018; Sartran et al., 2022). The second approach uses word-synchronous beam search (Stern et al., 2017) and approximates $p^{\mathcal{M}}$ by the set of inferred action sequences (Hale et al., 2018; Noji and Oseki, 2021; Yoshida et al., 2021), which we denote by $\tilde{p}^{\mathcal{M}}$. In this study, we focus on the latter approach since the former does not involve inference with the SLM model itself.

The process of word-synchronous beam search aims to model the joint prediction of the next token and its corresponding syntactic structure. For a token sequence x, the process can be represented by a sequence of sets of action sequences ending with SHIFT: $B_0, B_1, \ldots, B_{|x|}$. Here, B_i represents the set of (partial) syntactic structures in the beam when predicting token x_i, corresponding to the i-th step of word-synchronous beam search, and satisfying $B_i \subseteq A_i$. Note that $B_0 = \emptyset$, and each B_i is deterministically computed based on Algorithm 1. While previous work (Stern et al., 2017) introduces a word beam bottleneck, we instead limit the maximum number of actions between SHIFT actions to k_n to reduce inference time. The score function for selecting an action sequence $b'c$ is the joint probability:

$$\begin{cases} p^{\mathcal{M}}_{\text{joint}}(x_{<i}x_i, b'c), & \text{if } c == \text{SHIFT}, \\ p^{\mathcal{M}}_{\text{joint}}(x_{<i}, b'c), & \text{otherwise.} \end{cases}$$

5 Experiments

Evaluation. Here, we describe the overall flow of the experiments. For each treebank and strategy, we convert the gold trees to action sequences and train a base model $\mathcal{M}$ in a supervised manner. We then perform inference using word synchronous beam search with the trained model to obtain the set of action sequences $B_{|x|}$. We evaluate performance across a range of beam sizes, $k \in \{50, 200, 800\}$. To reduce inference time, we utilize fast-track selection with $k_s = k/50$ and limit the maximum number of actions between SHIFT actions to $k_n = 20$. For each setting, we train models with 3 different random seeds and report the average performance.

Algorithm 1 Word synchronous beam search with fast track selection and a step limit.

Input: $x_{\leq i}$ ▷ Token sequence
Input: $B = (k, k_s, k_n)$ ▷ Beam search params
Input: B_{i-1} ▷ Last beam
  $B'_i \leftarrow B_{i-1}$
  for $j = 1, \ldots$ **do**
    $C_{\text{fast}} \leftarrow \text{top}k_s(\{b' \cdot \text{SHIFT} \mid b' \in B'_i\})$
    $B_i \leftarrow B_i \cup C_{\text{fast}}$ ▷ Fast track selection
    $C \leftarrow \bigcup_{b' \in B'_i} \{b'c \mid c \in \mathcal{A}\}$
    $B'_i \leftarrow \text{top}k(C \setminus C_{\text{fast}})$ ▷ Select candidates
    for $b'c \in B'_i$ **do**
      if $c ==$ SHIFT **then**
        $B_i \leftarrow B_i \cup \{b'c\}$ ▷ Update beam
        $B'_i \leftarrow B'_i \setminus \{b'c\}$
    if $|B_i| = k \vee j \geq k_n$ **then**
      Break ▷ Quit search when the beam is full or the step limit is reached
  return B_i

Dataset. We use treebanks from 10 languages: English (Penn Treebank (Marcus et al., 1993)), Chinese (Chinese Treebank (Palmer et al., 2005)), French, German, Korean, Basque, Hebrew, Hungarian, Polish, and Swedish (SPMRL (Seddah et al., 2013)). Following Noji and Oseki (2021), we remove POS tags and split words into subwords. All evaluations in this paper are performed on the validation datasets. To reduce the size of the action set and simplify model training, we limit the n in NT(X;n) actions to a maximum of 10. To ensure consistent parsability across strategies, we restrict the train and validation data to instances where the gold trees are parsable by all strategies with $n \leq 10$. Furthermore, we only use sentences that are parsable with $n \leq 10$ and $k_n = 20$ for evaluation. Further details are provided in Appendix A.

Strategy. In our experiments, we analyze a total of 15 strategies: top-down, bottom-up, left-n-corner with $n \in \{1, 2, 3\}$, uniform-speculation with $\theta \in \{0.26, 0.35, 0.65, 0.74\}$, local-first with $h \in \{1, 2, 3\}$, and global-first with $d \in \{1, 2, 3\}$.[7] For simplicity, we consider the insertion position of NT actions at the subword level rather than the word level.

Model. For the model, we extend the commonly used syntactic language model the Recurrent Neural Network Grammar (RNNG) (Dyer et al., 2016)

[7] The values of θ are chosen such that i_v changes for a node v with $n_v = 2, 3, 4$ depending on θ.

Beam	English	Chinese	French	German	Korean
50	**BU** (88.7±0.3)	**LC-1** (86.1±0.2)	**TD** (81.8±0.2)	**LC-1** (86.4±0.1)	**BU** (84.5±0.1)
	LF-2 (87.3±0.1)	**BU** (86.1±0.1)	BU (79.6±0.1)	LC-2 (85.8±0.1)	LC-2 (84.1±0.1)
200	**LF-2** (89.4±0.1)	**LC-1** (87.0±0.2)	**TD** (83.3±0.2)	**LC-1** (87.3±0.1)	**BU** (84.5±0.1)
	LF-3 (89.4±0.0)	BU (86.6±0.3)	US-0.26 (81.1±0.1)	LC-2 (86.5±0.0)	LF-1 (84.2±0.1)
800	**TD** (90.9±0.1)	**LC-1** (87.0±0.2)	**TD** (83.7±0.2)	**TD** (87.7±0.1)	**BU** (84.4±0.1)
	LF-3 (90.2±0.0)	BU (86.7±0.2)	US-0.26 (81.8±0.1)	LC-1 (87.4±0.1)	LF-1 (84.2±0.1)

Beam	Basque	Hebrew	Hungarian	Polish	Swedish
50	**BU** (83.0±0.1)	**LF-1** (80.8±0.3)	**LC-1** (87.2±0.1)	**GF-1** (78.9±0.3)	**LC-1** (72.8±0.2)
	LF-1 (82.8±0.1)	LC-1 (80.5±0.3)	LC-2 (86.6±0.1)	BU (77.1±0.1)	US-0.26 (69.8±0.1)
200	**BU** (83.1±0.1)	**TD** (82.2±0.3)	**LC-1** (88.1±0.1)	**GF-1** (79.5±0.1)	**LC-1** (73.5±0.1)
	LC-1 (83.1±0.2)	LF-1 (81.6±0.3)	LC-2 (87.1±0.0)	BU (77.2±0.2)	TD (73.0±0.3)
800	**LF-1** (83.3±0.2)	**TD** (83.7±0.2)	**LC-1** (88.1±0.1)	**GF-1** (79.5±0.1)	**TD** (74.9±0.3)
	LC-1 (83.1±0.2)	LF-3 (82.3±0.2)	TD (87.9±0.1)	BU (77.0±0.3)	LC-1 (73.6±0.2)

Table 2: Top-2 strategies for the labeld parsing f1 scores for each dataset and beam size. TD and BU denote top-down and bottom-up strategies, and LC, US, LF, and GF denote left-n-corner, uniform-speculation, local-first, and global-first strategies with their corresponding parameters. Mean f1 scores and standard errors are shown in the parentheses.

to handle the proposed generalized shift-reduce action set. The implementation is based on the batched version of RNNG (Noji and Oseki, 2021). For the action set implementation, we simply represent SHIFT, REDUCE, and each NT(X;n) action by one-hot vectors. For each setting, we train a model for either 80 epochs or 8000 steps, whichever is larger, and evaluate the model with the lowest validation loss. Details of the training settings are provided in Appendix B.

5.1 Results on Parsing

First, we analyze parsing performance. We calculate the labeled F1 score using the highest-scoring action sequence in $B_{|x|}$. Table 2 shows the top two performing strategies for each language, and Figure 2 presents the parsing performance for all strategies. Note that in Figure 2, strategies are sorted from top-down to bottom-up from left to right for each speculation parameterization. The results reveal that the strategy that maximizes parsing performance depends on the language and beam size. For example, for English, bottom-up performs best when $k = 50$, local-first ($h = 2, 3$) performs best when $k = 200$, and top-down when $k = 800$. Similarly, top-down shows higher F1 scores than other strategies for French, German, Hebrew, and Swedish when $k = 800$. In contrast, for Chinese, Korean, and Basque, bottom-up, left-n-corner ($n = 1$), or local-first ($h = 1$) obtain higher F1 scores for all beam sizes. For these languages, the performance of top-down is lower compared to other strategies, especially when the beam size is small (Figure 2). The sentence probability marginalized over the beam, $\tilde{p}^{\mathcal{M}}$, showed a similar overall trend to the parsing performance. We show the results for $\tilde{p}^{\mathcal{M}}$ in Appendix C.

5.2 Results on Structure-conditioned Token Probability

Figure 3 shows the perplexity based on the $p^{\mathcal{M}}_{\text{token}}$ for the best action sequence obtained by beam search for English, Chinese, German, and Korean.[8][9] Generally, higher speculation leads to lower perplexity, i.e., higher $p^{\mathcal{M}}_{\text{token}}$, regardless of the speculation parameterization. However, for Chinese and Korean, Basque, Hungarian, and Polish, perplexity tends to be higher when the degree of speculation is too high when the beam size is smaller.[10]

6 Discussion

6.1 Is the Optimal Strategy Universal across Languages?

The results of the experiments suggest that the optimal strategy for incremental structure prediction in syntactic language models is not universal across

[8] This is different from the sentence probability $p^{\mathcal{M}}$.

[9] The results for other languages are shown in Appendix C.

[10] Basque, Hungarian, and Polish also show similar trend (Appendix C).

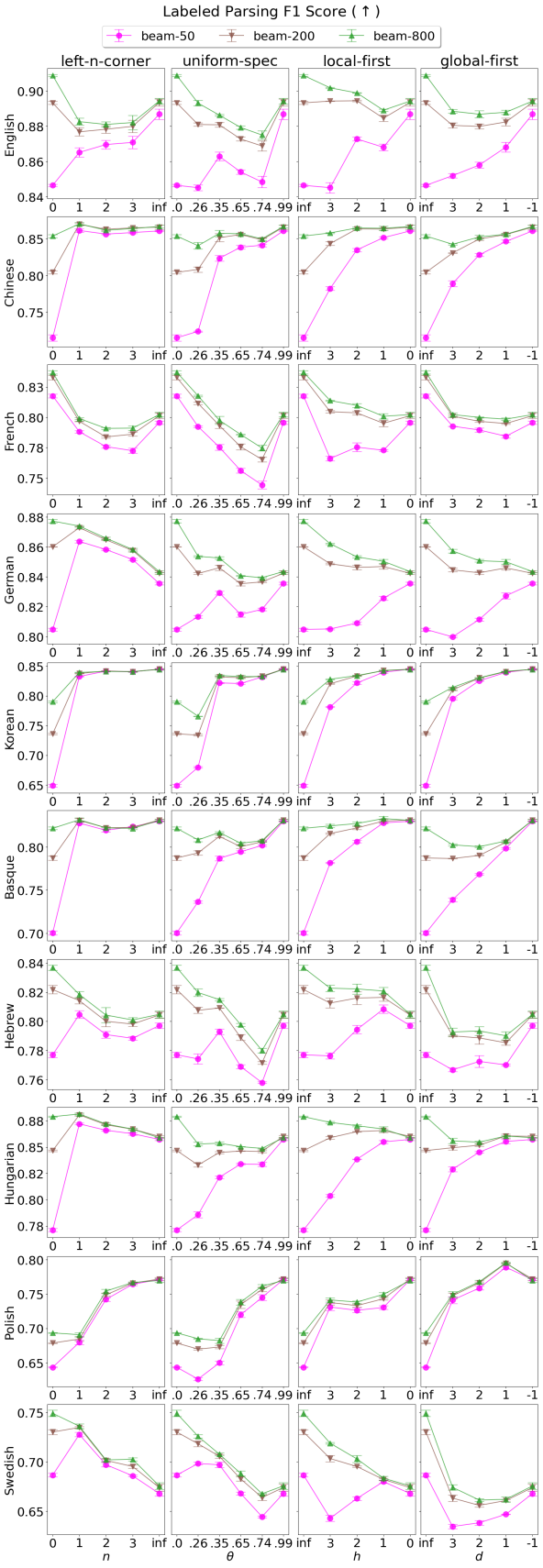

Figure 2: Labeled parsing F1 scores for all datasets. Error bars show the standard error of the mean.

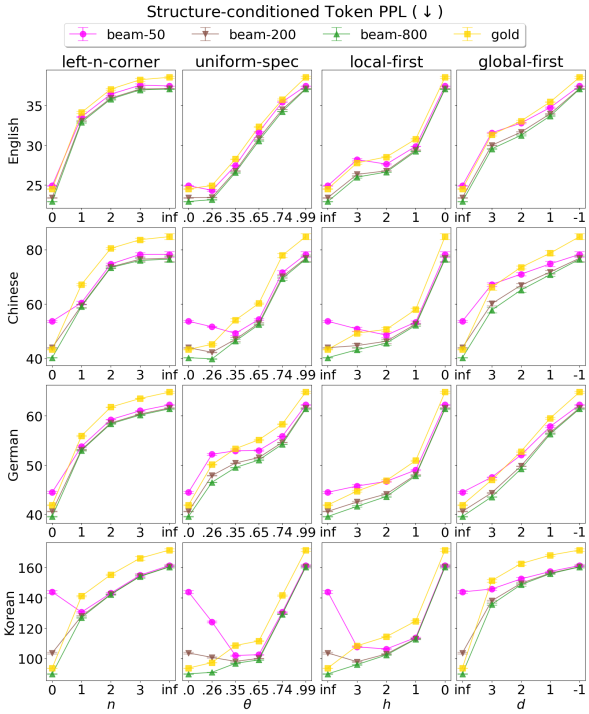

Figure 3: Perplexity based on $p^{\mathcal{M}}_{\text{token}}$. Error bars show the standard error of the mean.

languages, but rather language-specific. Previous research has suggested that left-corner is a better strategy due to its stack size efficiency, but our findings indicate that it is not necessarily the best in practical tasks.

What factors contribute to these differences between languages? If we simply consider the amount of information available during inference, less speculative strategies should be advantageous even with larger beam sizes. However, contrary to this expectation, top-down outperforms less speculative strategies in some languages. We hypothesize that this is due to a combination of two factors: the ease of learning of each strategy and the required parallel inference capacity.

First, Figure 4 shows the validation loss, i.e., negative joint log-likelihood $-\log p^{\mathcal{M}}_{\text{joint}}$, for English, Chinese, German, and Korean for the same data points as in Figure 2.[11] Generally across all languages except Korean, top-down has the lowest loss, followed by left-n-corner ($n = 1$), indicating that these strategies, especially top-down, are easier to learn.[12]

[11] The results for other languages are shown in Appendix C.

[12] For other strategies, except for global-first parameterization, we generally observe that lower speculation leads to better learning, i.e., lower validation loss. However, bottom-up sometimes shows lower loss than strategies other than top-down.

Second, top-down requires larger beam size, i.e., parallel inference capacity, than other less speculative strategies because top-down cannot use token information to predict structures. Furthermore, top-down requires even larger beam size for left-branching languages as discussed in the previous work (Abney and Johnson, 1991; Yoshida et al., 2021).

Overall, the top-down strategy exhibits a trade-off between ease of learning, which contributes to strong performance, and the difficulty of inference due to the required large beam size. The differences in the optimal strategy across languages might be attributed to differences in the balance of this trade-off. For example, in English, German, Hebrew, and Swedish, the parsing performance of top-down is low when the beam size is small, but it significantly improves as the beam size increases, becoming the best strategy at $k = 800$ (Figure 2). In Chinese and Korean, which are more left-branching than English, the performance of top-down tends to be lower than that of less speculative strategies like bottom-up, even with beam size $k = 800$.

6.2 Does syntactic structure contribute to token prediction?

In speculative strategies, token prediction is conditioned on the already-predicted syntactic structures. Thus, if $p^{\mathcal{M}}_{\text{token}}$ increases with the degree of speculation, i.e., the amount of structures usable for token prediction, syntactic structure is likely to be informative for token prediction. As shown in Figure 3, $p^{\mathcal{M}}_{\text{token}}$ tends to increase with the degree of speculation, suggesting that syntactic structure indeed captures information about tokens. For some languages, e.g., Korean, Chinese, Basque, and Polish, $p^{\mathcal{M}}_{\text{token}}$ decreases for more speculative strategies, likely due to inference failure. Nevertheless, with gold actions, $p^{\mathcal{M}}_{\text{token}}$ increases with the degree of speculation across all languages, which also supports the informativeness of syntactic structures.

Meanwhile, for all languages and strategies, the token probability conditioned on the gold tree is lower than that conditioned on the structures inferred by the model. This result suggests that, from the perspective of token prediction, a certain level of parsing accuracy is sufficient, and exact parsing may not be necessary. In fact, it is also argued that human language processing only utilizes partial shallow structures (Sanford and Sturt, 2002; Ferreira et al., 2002; Ferreira and Patson, 2007), and Noji and Oseki (2023) showed that syntactic

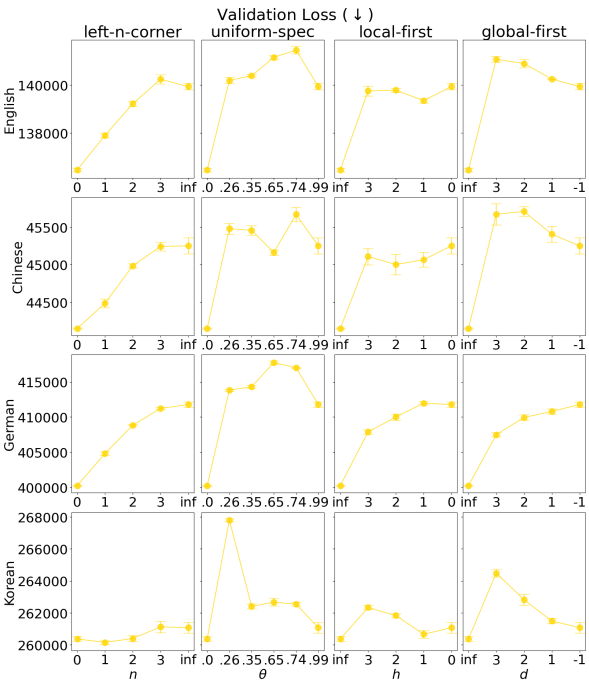

Figure 4: Validation loss, i.e., $-\log p^{\mathcal{M}}_{\text{joint}}$. Error bars show the standard error of the mean.

ablation, i.e., removing some syntactic categories, improves the syntactic generalization ability of top-down models in English. Therefore, to further investigate the extent to which syntax is necessary for token prediction, it would be necessary to perform syntactic ablation across various strategies in future work.

7 Conclusion

This study analyzed whether the incremental structure prediction process in natural language is universal across languages or language-specific. We considered a total of 15 strategies based on 4 different parameterizations of speculation. Experiments on 10 typologically diverse languages suggest that the optimal strategy can vary across languages and is influenced by two factors: the ease of learning and the required parallel inference capacity. Furthermore, a comparison between strategies with different degrees of speculation reveals that the syntactic structure of natural language is indeed informative for token prediction, while also suggesting that exact parsing might not be necessary. Finally, this study focused on phrase structure; however, natural language also encompasses other structures such as dependency and semantic structures. Future work examining strategies for such structures is expected to further reveal universals and differences across languages.

References

Steven P Abney and Mark Johnson. 1991. Memory requirements and local ambiguities of parsing strategies. *J. Psycholinguist. Res.*, 20(3):233–250.

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.

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.

Fernanda Ferreira, Karl G D Bailey, and Vittoria Ferraro. 2002. Good-enough representations in language comprehension. *Curr Dir Psychol Sci*, 11(1):11–15.

Fernanda Ferreira and Nikole D Patson. 2007. The 'good enough' approach to language comprehension. *Lang. Linguist. Compass*, 1(1-2):71–83.

John Hale, Chris Dyer, Adhiguna Kuncoro, and Jonathan Brennan. 2018. Finding syntax in human encephalography with beam search. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2727–2736.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 9(8):1735–1780.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

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.

Tatsuki Kuribayashi, Ryo Ueda, Ryo Yoshida, Yohei Oseki, Ted Briscoe, and Timothy Baldwin. 2024. Emergent word order universals from cognitively-motivated language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 14522–14543, Stroudsburg, PA, USA. 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*.

Hiroshi Noji and Yusuke Miyao. 2014. Left-corner transitions on dependency parsing. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 2140–2150, Dublin, Ireland. Dublin City University and Association for Computational Linguistics.

Hiroshi Noji and Yohei Oseki. 2021. Effective batching for recurrent neural network grammars. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 4340–4352, Online. Association for Computational Linguistics.

Hiroshi Noji and Yohei Oseki. 2023. How much syntactic supervision is "good enough"? In *Findings of the Association for Computational Linguistics: EACL 2023*, pages 2300–2305, Dubrovnik, Croatia. Association for Computational Linguistics.

Martha Palmer, Fu-Dong Chiou, Nianwen Xue, and Tsan-Kuang Lee. 2005. Chinese treebank 5.1 ldc2005t01u01.

Peng Qian, Tahira Naseem, Roger Levy, and Ramón Fernandez Astudillo. 2021. Structural guidance for transformer language models. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 3735–3745, Online. Association for Computational Linguistics.

Philip Resnik. 1992. Left-corner parsing and psychological plausibility. In *COLING 1992 Volume 1: The 14th International Conference on Computational Linguistics*.

Anthony Sanford and Patrick Sturt. 2002. Depth of processing in language comprehension: not noticing the evidence. *Trends Cogn. Sci.*, 6(9):382.

Laurent Sartran, Samuel Barrett, Adhiguna Kuncoro, Miloš Stanojević, Phil Blunsom, and Chris Dyer. 2022. Transformer grammars: Augmenting transformer language models with syntactic inductive biases at scale. *Transactions of the Association for Computational Linguistics*, 10:1423–1439.

Djamé Seddah, Reut Tsarfaty, Sandra Kübler, Marie Candito, Jinho D Choi, Richárd Farkas, Jennifer Foster, Iakes Goenaga, Koldo Gojenola, Yoav Goldberg, Spence Green, Nizar Habash, Marco Kuhlmann, Wolfgang Maier, Joakim Nivre, Adam Przepiórkowski, Ryan Roth, Wolfgang Seeker, Yannick Versley, Veronika Vincze, Marcin Woliński, Alina Wróblewska, and Éric Villemonte de la Clergerie. 2013. Overview of the SPMRL 2013 shared task: A cross-framework evaluation of parsing morphologically rich languages. In *Proceedings of the Fourth Workshop on Statistical Parsing of Morphologically-Rich Languages*, pages 146–182.

Mitchell Stern, Daniel Fried, and Dan Klein. 2017. Effective inference for generative neural parsing. In *Proceedings of the 2017 Conference on Empirical*

Methods in Natural Language Processing, pages 1695–1700, Copenhagen, Denmark. Association for Computational Linguistics.

Yushi Sugimoto, Ryo Yoshida, Hyeonjeong Jeong, Masatoshi Koizumi, Jonathan R Brennan, and Yohei Oseki. 2024. Localizing syntactic composition with left-corner recurrent neural network grammars. *Neurobiology of Language*, 5(1):201–224.

Ryo Yoshida, Hiroshi Noji, and Yohei Oseki. 2021. Modeling human sentence processing with left-corner recurrent neural network grammars. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 2964–2973, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.