On the relative impact of categorical and semantic information on the induction of self-embedding structures

Antoine Venant
OLST, Université de Montréal
antoine.venant@umontreal.ca

Yutaka Suzuki
OLST, Université de Montréal
yutaka.suzuki@umontreal.ca

Abstract

We investigate the impact of center embedding and selectional restrictions on neural latent tree models' tendency to induce self-embedding structures. To this aim we compare their behavior in different controlled artificial environments involving noun phrases modified by relative clauses, with different quantities of available training data. Our results provide evidence that the existence of multiple center self-embedding is a stronger incentive than selectional restrictions alone, but that the combination of both is the best incentive overall. We also show that different architectures benefit very differently from these incentives.

1 Introduction

Grammar induction is the task of inducing hierarchical syntax trees from *indirect* observations, most often provided by the string yields of those trees (raw sentences)[1]. The most common type of approach parametrizes a joint distribution over (observed) strings and (latent) trees, and fit it to the data by optimizing a language modeling objective, *i.e.* minimizing the cross entropy between the model's marginal distribution over strings and the observed distribution. We will refer to approaches of this kind as *latent tree models*. Latent tree models are interesting because they can provide distributional evidence for (or counter arguments to) the structures stipulated by linguistic theories, help investigate inductive biases and language model pretraining, or build bridges between neural models and symbolic ones.

Grammar induction, however, is a rather difficult task. For the longest time, models were mostly trained and tested on very short sentences of about 10 words, struggled to beat baselines such as right- or left-linear grammars (Carroll and Charniak, 1992), heavily relied on heuristics for clustering (Clark, 2001), initialization (Klein and Manning, 2001, 2002, 2004) and/or assumed part-of-speech tagged inputs (Bisk and Hockenmaier, 2013). Whereas several potential culprits like an ill-shaped objective function (Klein and Manning, 2001), or data quantity (Pate and Johnson, 2016) have been named, a comprehensive explanation of the underlying difficulties is still lacking.

More recently, neural grammar induction models surfaced (Shen et al., 2018; Htut et al., 2018; Shen et al., 2019; Kim et al., 2019b,a; Yang et al., 2021; Zhu et al., 2020), which substitute the discrete features of their predecessors with continuous representations of input words and models' states, and parametrize the joint probability over tree structures and strings using a neural network. While these models considerably improved the state of the art for phrase structure induction from words alone, there remains a large gap between their performances and those of supervised parsers. An important question is **why a language modeling objective would align well with some (let alone all) of the intended structural patterns**. Moreover, the answer to this question might well be negative, because better unsupervised parsers are often worse language models and *vice versa* (Kim et al., 2019a).

In hope to improve our understanding of the matter, this paper investigates the combinations of training signal and neural models able to induce self-embedding structures, and the generalization capabilities of the learned models to larger phrases. We focus on the case of noun phrases, whose linguistic analysis is tied to phenomena such as (long-distance) subject-verb agreement, commonly used, across different languages, in benchmarking language models' syntactic awareness (Linzen et al., 2016; Marvin and Linzen, 2018; Li et al., 2023). We are specifically interested in the relative impact

[1] At least in one of its common usages in machine learning. Other usages include the task of inferring a specific form of formal grammar, or a recognizer from example sentences of a language.

Proceedings of the Second Workshop on the Bridges and Gaps between Formal and Computational Linguistics (BriGap-2), pages 79–96

of **categorical** incentives based on the sequences of coarse-grained part of speech categories, and lexical incentives based on more fine-grained **semantic** distinctions between words within a given category.

Since the complexity of natural language makes it hard to study specific aspects of the training signal and output structures in isolation, we experiment with artificial data generated with probabilistic grammars. This allows us to control for the presence of different incentives in the training signal, as well as the quantity of available training data. It also guarantees that at least one optimal model exists which leverages the intended structures. Unlike other works evaluating grammar induction systems on formal languages (Lari and Young, 1990; Lan et al., 2022), we focus on the *strong* learning of the intended structures, and use larger grammars with a sizeable lexicon to make learning syntactic categories and representing lexical features an integral (and non-trivial) part of the task. We do not consider alternative objectives (such as Minimum Description Length), because we precisely want to assess to what extent the (currently) more scalable language modeling objective aligns with theoretical patterns, if it does.

In §2 we discuss two incentives for inducing a self-embedding analysis of noun-phrases and relative clauses (henceforth, RC). §3 then presents how these incentives are implemented into four different artificial training signals. §4 details the experimental setup leveraging these data, and §5 discusses our findings and conclusions.

2 Why would a language model build noun phrases?

2.1 Distributional considerations

Linguists argue (across a variety of languages) that a noun, like *people*, can merge with a restrictive modifier, like *in a blue shirt* to form a noun phrase (NP), like *people in a blue shirt* (*e.g.* Baker, 1995; Tellier, 2003, for English and French, respectively). This would for instance happen twice when forming the English sentence *these* $[_{\text{NP}}$ *people in a* $[_{\text{NP}}$ *blue shirt with a collar*]] *are staff members*.

This analysis is often justified by a similarity in distribution between longer sequences (*people in a blue shirt*) and shorter ones (*people*). For instance, both are good candidates to fill the blanks in the following context: *these __ are staff members*.

It is thus compelling to assume that *people in a blue shirt with a collar* forms a constituent which inherits the morphosyntactic features (in particular, the number) and combinatorial properties of its head (*people*), because it explains why it combines, and agrees, with verbs as the bare noun *people* does. An important contribution of the hierarchical structure to that argument, is that it brings heads and dependents (the verb and subject in the above example) closer by grouping the intervening material inside a substructure who contributes little to the purpose of predicting agreement or the surounding context, and can therefore be pruned[2]. Formally, this process can (for instance) be equivalently articulated in a dependency framework, or in a headed constituency framework (Eisner and Satta, 1999; Nederhof and Satta, 2011).

2.2 An expected empirical difficulty

We have however no theoretical insurance that considerations like the above are sufficient for latent tree models to succeed. An important problem is raised by Klein and Manning (2002): the distribution of contexts in which a sequence occur is not necessarily a good indicator as to whether it is a constituent or not. Consider these two sequences:

A *the student who frequently questions the professor caused a problem*

B *the professor caused a problem*

A and B could plausibly occur in a lot of similar contexts, in which there are indeed constituents of the same type[3]. However, considered as a subsequence of A, B is not a constituent under any linguistic standard.

The problem is emphasized if one expects models to leverage a rather coarse notion of syntactic category (such as POS), because we can easily imagine such models to learn a **right-linear** grammar with a rule $S \rightarrow$ `det noun who verb` S. This grammar would generate sentences and parses such as $[_S$ *The student who questions* $[_S$ *the professor who questions* $[_S$ *the professor caused a problem*]]]. While we might find the induced string language somewhat reasonable, treating RCs as embedding sentence types is linguistically very unconventional.

[2] Or receive less attention, in a more relaxed, continuous vision of sentence processing.

[3] For instance in the context *Do you know whether ... ?*: *C*:*Do you know whether the student who frequently questions the professor caused a problem? D*:*Do you know whether the professor caused a problem?*.

2.3 Two possible incentives

Of course, an accurate language model needs to leverage more than categorical information. Thus, we might hope that semantic concerns, such as **selectional restrictions** (or, selectional *preferences* in a probabilistic view), can help break unwanted symmetry. For instance, the respective contexts of A': *the student who eats a cookie with sugar sprinkles passed the test* and B': *a cookie with sugar sprinkles passed the test* are probably more distinguished than those of A and B since, unlike B, B' is very unlikely to appear as a standalone sentence, as *cookie* violates the selectional restrictions of the verb *pass*. If a model is able to find a middle ground between relying on purely categorical and semantic knowledge, then we can hope that i) it makes a symmetric treatment of A and A' (based on categorical similarity) and ii) it makes the expected analysis of A' (based on semantic knowledge), hence of A (based on i). Moreover, selectional restrictions introduce a rich diversity of long-distance dependencies (such as between *student* and *pass* in A'), thus reinforcing the linguistic argument developed above.

Independently, it has been argued (Chomsky, 1956; Partee et al., 1990) that the set of 'grammatical' sentences of many languages are not representable by a right- or left-linear grammar because it involves **unbounded center-embedding** (or equivalently, unbounded well-nested projective dependencies). In English or French, this can for instance be argued through self-embedding of object RCs: *the student [that the professor [that your friend [(that . . .)] had]] dislikes] passed the test*. The set of sentences of this form is related[4] to the formal language $\{a^n b^n \mid n \in \mathbb{N}^*\}$, a canonical example of non-regular set. The possibility that this empirically affects latent tree models might however seem more remote, because it rests on an abstract notion of competence and an infinite set of sentences of arbitrary complexity, most of which are not attested (Karlson, 2007) or are rejected by speakers (Christiansen and MacDonald, 2009). Nevertheless, levels of center-embedding below four are attested (Karlson, 2007). Jin et al. (2018) introduced a grammar-based system that outperformed its contemporary competitors on English unsupervised parsing, and found that their system also achieved better performances on synthetic data with bounded center embeddings. This suggests that the ability to infer such bounded depth center-embedding, even on synthetic data with a very small lexicon, might be important for achieving good unsupervised parsing performances on natural language benchmarks. Moreover, state of the art neural language models' architectures have been theoretically and empirically demonstrated able to learn such bounded-depth occurrences (Yao et al., 2021).

Based on the above discussion, our objective will therefore be to assess the respective effect of **selectional restrictions** and **multiple center embedding** on different latent models.

3 Artificial Data

3.1 Target self-embedding

Our experiments are built around examples of self-embedding provided by French noun-phrases modified by subject, or object RC. These structures feature two types of self-embedding: final self-embedding (when the embedding constituent does not yield any material to the right of the embedded constituent), and center self-embedding (when the embedding constituent yields material to the left and right of the embedded constituent). In Figure 1, [$_{NP}$ *journaliste qui cherche le* [$_{NP}$ *succès*]] illustrates a case of final self-embedding, whereas [$_{NP}$ *article que le journaliste qui cherche le* [$_{NP}$ *succès*] *écrit*] illustrates a case of center self-embedding. In particular, subject and object RC respectively involve final and center self-embedding of NP. Additionally, RCs nested within object RC involve center self-embedding of CP and RCs nested within subject RCs involve final self-embedding of CP.

3.2 Four types of training signals

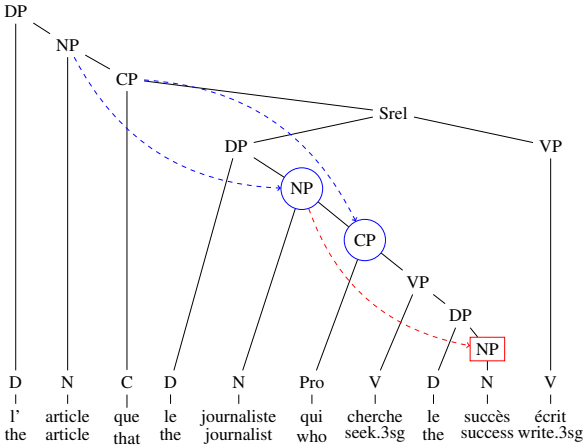
Our training data is artificially generated using PCFG (Probabilistic Context-Free Grammar). We first describe the (ideal) defining properties of the different training signals compared in our experiments, postponing the implementation's specifics to §3.3. All signals involve simple transitive or oblique sentences, with exactly one direct or exactly one oblique object. Every noun can be modified by exactly one RC which ensures that the data features examples of self-embedding.

This common basis is declined into four different

[4] Formally: $a^n b^n$ is an homomorphic image of the considered set of SVO sentences with arbitrary nesting of relative object in the subject position, and it follows from well-known closure properties that the former is regular (and thus represented by a right- or left-linear grammar) only if the latter (proven not regular) is.

-sr -mce	-sr +mce	+sr -mce	+sr +mce	sentence
1	1	1	1	le projet qui intéresse le journaliste qui écrit l'article présente un progrès 'the project which interesses the journalist who writes the article displays progress'
1	1	0	0	le projet qui parle au journaliste qui mange l'article regarde un progrès 'the project which talks to the journalist who eats the article watches progress'
0	1	0	1	le projet auquel le journaliste qui écrit l'article contribue présente un progrès 'the project to which the journalist who writes the article contributes displays progress'
0	1	0	0	le projet auquel le journaliste qui mange l'article parle regarde un progrès 'the project to which the journalist who eats the article talks watches progress'

Table 1: Type of sentences and compatibility with each configuration.



'the article that the journalist who seeks success writes'

Figure 1: French example with phrase structure tree and English gloss. Dashed edges indicate self-embedding, blue circles center embedded constituents and red squares final embedded constituents.

signals, depending on whether or not we simulate the two incentives discussed in the previous section. **-sr** means that the data does not simulate **selectional restrictions** in the probabilistic sense that all verbs, prepositions and nouns are conditionally independent of other words given their POS. This condition should thus assign equal probability to *article* (article) and *journaliste* (journalist) as subject of *écrire* (write). **+sr**, in contrast, means that the verb more likely selects semantically plausible subjects and objects. **+/-mce** means that the data exhibit / does not exhibit **multiple center-embedding**, according to whether noun phrases can be modified by either an (oblique) object RC or subject RC / can only be modified by a subject RC. Remark that the **-sr** and **+mce** configurations respectively *allow* selectional preference violations and object RCs, but do not *require* these to occur in every sentence. Thus, one may still observe some semantically sound sentences in the **-sr** configurations if appropriate verbal arguments are randomly selected (though, this will be unlikely if there are more lexical items violating a verb's selectional preferences than items respecting them), as well as sentences without multiple center-embeddding in the **+mce** configuration. Table 1 presents four example sentences and their compatibility with the four configurations: the first sentence neither violates selectional preferences nor features multiple center-embedding, hence could be observed under all four training configurations. The second sentence violates selectional preferences, and should thus only be observed in the two **-sr** configurations. The third sentence does not violate selectional preferences, but has multiple clausal and NP center-embedding due to the oblique object RC. It therefore can be observed only in the two **+mce** configurations. The fourth sentence both violates selectional preferences and has multiple center-embedding, and can therefore only be observed in the **-sr+mce** configuration. Note however, that table 1 presents a somewhat idealized picture: in practice, because the lexicon is acquired semi-automatically and the data generated probabilistically (see §3.3), selecting an inappropriate verbal argument will be unlikely rather than strictly impossible in the **+sr** configurations.

In line with the discussion in §2.2, we expected **-sr-mce** to provide the weakest signal for the induction of self-embedding structures and the condition **+sr+mce** to provide the strongest incentives.

3.3 Data generation

We used probabilistic context free grammars to generate the data for each configuration. More specifically, we used *lexicalized* PCFGs (Nederhof and Satta, 2011, henceforth LPCFG) to encode selectional preferences. Rather than the standard bilexical LPCFG, we used trilexical ones, with up to two anchors per nonterminal (this allows to keep the verb, preposition and object interdependent in oblique object constructions). We handcrafted *delexicalized* rules with anchor variables like the

following (used for the generation of a preposition x_1 and oblique object x_2, conditionally to a verb x_0):

$$\text{Vobl}'_{\langle x_0 \rangle} \mapsto \text{Vobl}_{\langle x_0 \rangle} \text{PP}_{\langle x_1, x_2 \rangle} [\text{o\_obj}(x_1, x_2 \mid x_0)] \quad (1)$$

In this example, the symbols Vobl', Vobl, PP are called delexicalized nonterminals and the symbols x_0, x_1 and x_2 are variables used as placeholders for terminal symbols. The expression $\text{o\_obj}(x_1, x_2 \mid x_0)$ is an *abstract weight*, formally representing a function associating a real-valued weight to concrete values for the variables x_0, x_1 and x_2 (in this case, the joint conditional probability of preposition and object, given verb).

To generate data, a delexicalized grammar needs to be combined with a set of terminal symbols, and **concrete** functions instantiating the abstract weights (henceforth, a *lexicon*). This allows to turn each delexicalized rule into a set of concrete rules. For instance, assuming that $\text{o\_obl}(\text{to}, \text{journalist} \mid \text{talk}) = 0.3$ and $\text{o\_obl}(\text{about}, \text{project} \mid \text{talk}) = 0.7$, the rule in (1) would yield the following *lexicalized* rules:

$$\text{Vobl}'_{\langle \text{talk} \rangle} \mapsto \text{Vobl}_{\langle \text{talk} \rangle} \text{PP}_{\langle \text{to,journalist} \rangle} [0.3] \quad (2)$$

$$\text{Vobl}'_{\langle \text{talk} \rangle} \mapsto \text{Vobl}_{\langle \text{talk} \rangle} \text{PP}_{\langle \text{about,project} \rangle} [0.7] \quad (3)$$

The resulting LPCFG can then be used to generate sentences with both a constituency structure and a dependency structure[5].

The remaining difficulty is to craft a lexicon sufficiently large for models to be 'forced' into some kind of categorization, while reasonably simulating the `+sr` configuration. To achieve this, we leveraged CamemBERT (Martin et al., 2020), a French masked language model. We manually fixed the sets of functional categories (prepositions $\mathcal{P}$ and determiners $\mathcal{D}$), as well as two sets of 34 transitive verbs and 20 intransitive oblique verbs. We then bootstrapped a lexicon of nouns and probability distributions using CamemBERT. To this effect, we made a set of requests to the masked language model. Let us examplify this with *parler* (talk). To obtain both oblique object ($\text{o\_obl}(x_1, x_2 \mid \text{parler})$ above) and subject probabilities, we used a set of masked requests of the form

(4) d_1 <mask>$_s$ parle x_1 d_2 <mask>$_o$

where $d_{1/2}$ range over determiners and x_1 over prepositions. For instance, $d_1 =$ *un*, $x_1 =$ *à* and $d_2 =$ *la* corresponds to the masked request:

(5) un <mask>$_s$ parle à la <mask>$_o$
a.m <mask>$_s$ talk.prs.3sg to the.f <mask>$_o$

For each request, CamemBERT outputs two conditional distributions $P_{s/o}(x_2 \mid d_1, x_1, d_2, \text{parler})$, one for each of the two masked positions (subject and object). From there, we simply assumed uniform prior and marginalized over any extra variable (like the determiners). For instance, we obtained $\text{o\_obl}(x_1, x_2 \mid \text{parler})$ by computing $\frac{1}{|\mathcal{D}|^2|\mathcal{P}|} \sum_{d_1,d_2} P_o(x_2 \mid d_1, x_1, d_2, \text{parler})$. We proceeded similarly for the other verb-noun distributions, then performed some additional filtering, keeping only the top 100 nouns for each conditional distribution, and only nouns that are both subject and object of some verbs (to ensure that they support both kind of modification by RC), and finally re-normalizing. After normalization, the resulting sentences are inflected using the French and English surface realizer `PyRealB` (Molins and Lapalme, 2015; Lapalme, 2020).

To simulate the four configurations, we combined the same delexicalized grammar with different concrete weights. The above procedure yielded a lexicon of 951 nouns, as well as the verb selectional distributions for the `+sr` configuration. For the `-sr` configuration, we replaced these distributions with uniform distributions on the relevant domains. In both cases, we used a uniform distribution for the choice of the main verb, and relied on Bayes' theorem to generate the verb in object and subject RC depending on the modified noun. In all configurations, we set a fixed probability (0.3) for modifying each noun. The nesting of RCs on a given noun thus follows a geometric law, and the number of generated sentences decays exponentially with the depth of nested RCs. Each training dataset therefore contains very rare instances of deeply nested embeddings. The grammar used for `+mce` condition has equal chances of attaching an object and subject RC, while the one for `-mce` only attaches subject RC. The probability of attachment and the expected degree of nesting are controlled and remain the same across configurations.

Note that, in the `-sr` configurations, the lexical anchors of the LPCFG can be safely deleted without changing the language (the anchors' only purpose is to implement selection restrictions). This operation leaves grammars in Chomsky Normal

[5] For dependency structures, a few adaptations are needed from the bilexical to trilexical case. Since the paper focuses on constituency, we do not expand on these technical matters.

Form with less than 30 nonterminal symbols. In contrast, in the +sr configurations, the generating grammars have over 17000 nonterminals, most of which are probably[6] necessary. In addition, these conditions involve long-distance dependencies between (at least) the subject of the main clause and the main verb. Note also, that both **-mce** configurations make it theoretically possible to perfectly fit the ground-truth distribution with a right-linear PCFG or a left-linear PCFG whereas the **+mce** conditions theoretically require branching structures for a perfect fit.

We might question whether our implementation of the +sr condition matches its definition, given the semi-automatic acquisition of the lexicon. However, looking at the verb-argument distributions, we found that the top subject and objects are generally semantically sound. For instance, the top 5 subjects of *investir* (*invest*) are *groupe* (*group*), *banque* (*bank*), compagnie (*company*), region (*region*) and *ville* (*city*) (covering about 75% of the probability mass), while the top 5 subject of *eat* are *femme* (*woman*), *chien* (*dog*), *homme* (*man*), *fille* (*girl*) and *chat* (*cat*) (covering about 40% of the probability mass). We also checked that the overall distribution of nouns follows Zipf's law, and estimated the mutual information between subject and main verb from the generated data to be approximately 2 bits (against 0 in -sr configurations).

4 Experiments

4.1 Training and test data

Using the procedure described in § 3.3, we generated over one million sentences for each of the four configurations and removed duplicates. The remaining data were split into training and development sets. From each training set, we constructed four subsets of approximately 3k, 12k, 100k, and 400k sentences by recursive halving, ensuring that all smaller subsets are prefixes of the larger ones (e.g., the first 3k sentences appear in all four datasets). This resulted in 16 training sets and four development sets of 3k sentences each.

Since the models trained under -mce never observe object RC, it would be unfair to compare their ability to parse sentences with object RC to model which have. We thus mainly compare models on their performance on data from the -mce configuration, *i.e.* their ability to analyze noun phrases modified by subject RC. We use the +sr-mce configuration for evaluation, because none of the four configurations are biased against any of its sentences: sentences from the +sr-mce are as likely as any other sentence with the same sequence of POS to occur in the -sr training data, whereas the converse is not true. We therefore generated an (out-of-domain) test containing 5000 sentences for each sentence length up to 23 words (this corresponds to a maximal nesting depth of four RC), for a total of 75000 test sentences.

4.2 Models

We experimented with three strong neural latent constituency tree baselines: Neural PCFG and Compound PCFG (Kim et al., 2019a, henceforth, NPCFG and CPCFG) and Unsupervised Recurrent Neural Networks Grammars (URNNG, Kim et al., 2019b). Since the full parametrization of these three models would take too much space, we refer the interested reader to the original papers and recall only their most salient features.

NPCFG and CPCFG These models are based on a neural parametrization of PCFGs in Chomsky normal form. Both assume the number of nonterminal symbols of the grammar to be fixed as a hyperparameter. To generate a sentence, these models start from a designated nonterminal symbol S and recursively apply rewrite rules of the form $X \to \sigma$, replacing some nonterminal X with a sequence σ of one or two (terminal or nonterminal) symbols, until there remains only terminal symbols. The difference between NPCFG and CPCFG lies in how they model the choice of a rewrite rule. NPCFG parametrizes the probability of rewriting X with $X \to \sigma$ as proportional to $e^{f(u_X, v_\sigma)}$ where u_X and v_σ are learned embeddings for the left-hand side and right-hand side of the rule, and f is a neural network. CPCFG aims at weakening the context-free assumptions by making each step in the derivation dependent on a global context vector z. It thus generates z from a gaussian prior before applying the rewriting process. z is then shared between every rewriting decision, and the probability of $X \to \sigma$ becomes proportional to $e^{f(u_X \cdot z, v_\sigma)}$ where $\cdot$ is vector concatenation.

URNNG URNNG does not involve a discrete space of symbols and rules. It relies instead on a transition-based system inspired from shift-reduce parsers. A sentence is generated by successively applying SHIFT or REDUCE actions to a stack of

[6] It is hard to give a lower bound since PCFG minimization is undecidable.

tree fragments, until an end-of sentence symbol is generated. SHIFT generates a word and moves it on top of the stack, while REDUCE merges the two top elements of the stack into a single tree fragment. At every step, the model also maintains a stack of hidden states. The choice of the next action is parametrized as a function of the top-element of the stack of hidden states, and each action updates the stack of hidden states using a stack recurrent neural network (Kuncoro et al., 2017).

The tested models thus have important differences: URNNG, unlike the two others, generates words in a strict left-to-right order and does not involve symbolic rules. NPCFG is the only model whose decisions depend only on a **finite** set of configurations. CPCFG and URNNG stand on opposite sides of the parsing/language modeling tradeoff, with NPCFG and CPCFG achieving better parsing performance but much worse perplexity than URNNG on natural language. Unlike NPCFG, CPCFG and URNNG most likely have expressive capabilities beyond PCFG, though this has (to our knowledge) not be formally established. In particular, URNNG's use of stack LSTM suggests (Merrill et al., 2020; Weiss et al., 2018) that it could model data from the `+mce` conditions without branching structures, which lies beyond PCFG's strong expressive power[7]. Finally, the models use distinct strategies to marginalize over latent trees and estimate the probabilities of sentences: NPCFG and CPCFG use dynamic programs[8] whereas URNNG uses REINFORCE with control variate.

We tested the three models under the best set of hyperparameters[9] respectively reported in Kim et al. (2019a) and Kim et al. (2019b), using the authors' orginal implementation. In particular, we used CPCFG with 30 nonterminal symbols and 60 preterminal symbols, which means that, for the `-sr` configuration, a perfect language model and parser lies within the searched class of CPCFG models (cf §3.3), hence, that perfect parsing accuracy can be achieved on the test set.

We trained four instances of each model (CPCFG, NPCFG and URNNG) on a machine with a single RTX4090 GPU. Each run was initialized with a different random seed, on every training dataset. Training duration was controlled by the number of steps, to enable fair comparisons across datasets of different sizes. We trained for a maximum of 10^5 steps, performing validation every 500 steps, stopping early when perplexity on the development set failed to improve for three consecutive validations. Early stopping always triggered before the maximum number of steps was reached. Detailed hyperparameters are provided in Appendix A, and selected training, development and testing datasets, code and parsed test data are available at `https://github.com/suzuyuta/BriGap-2_2025`.

4.3 Metrics

We measured performances according to sentence-level (unlabeled) **F1 score**, the standard metrics for unsupervised constituency parsing. However, since our training data is designed around specific incentives for self-embedding of NP and CP, it does not reflect common linguistic arguments *e.g.* the positioning of the subject above VP, which are nevertheless evaluated by F1. We therefore report the recall on constituents of type NP and CP.

5 Results and discussion

Figure 2 presents the results achieved by the three models under the four configurations and four data sizes. The reported numbers are the mean performance and standard deviation across all four runs. Detailed results for each run are available in the appendix B (Table 3, 4, and 5).

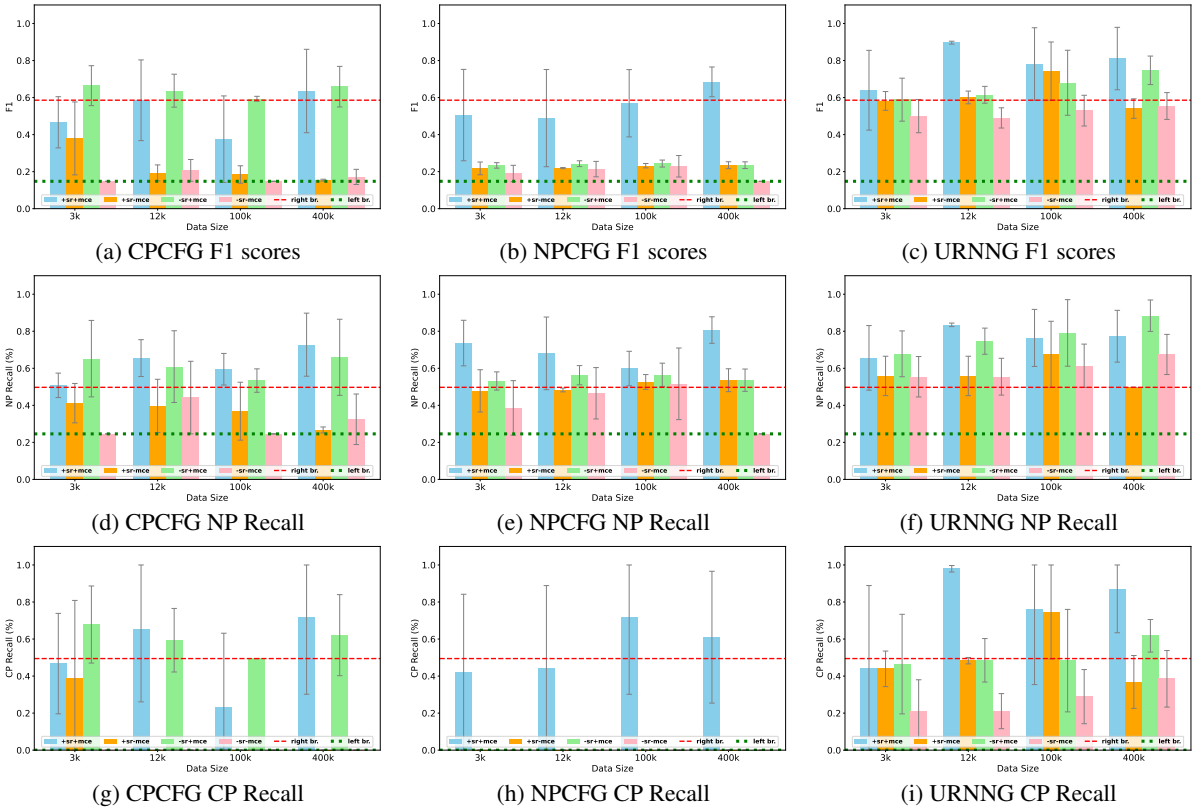
Overall impact of `sr` and `mce` The `+sr+mce` configuration (light blue) achieves the best results across the board, and the `-sr-mce` (pink) the worst results, confirming intuitions. The `-sr+mce` (light green) configuration tends to yield better result than the `+sr-mce` (orange), though there are exceptions for URNNG. This suggests that, to the extent that these phenomena are captured in our data, latent tree models are more sensitive to multiple center embedding than selectional restrictions. Furthermore, `+sr` configulations had the effect of increasing the number of preterminal symbols used in NPCFG and CPCFG models.

Model comparison URNNG seems to consistently achieve either comparable, or better performance than CPCFG, across all data sizes and configurations. This is rather surprising: for English, Kim et al. (2019a) reports a performance of 60.1

[7] URNNG essentially reduces to a standard LSTM language model when operating on linear tree structures.

[8] CPCFG uses variational inference to marginalize over z.

[9] Hyperparameters mainly involve embedding dimensions, hidden state dimensions, number of preterminals and nonterminals (when relevant), and number of KL annealing step for variational inference in URNNG.



(a) CPCFG F1 scores (b) NPCFG F1 scores (c) URNNG F1 scores

(d) CPCFG NP Recall (e) NPCFG NP Recall (f) URNNG NP Recall

(g) CPCFG CP Recall (h) NPCFG CP Recall (i) URNNG CP Recall

Figure 2: Sentence-level F1, NP and CP recall for CPCFG and URNNG. Dashed lines show the right-branching (red) and left-branching baseline scores (green)

for the best PCFG model against a performance of 52.6 for the best URNNG model on the Penn Treebank (Marcus et al., 1993). Hence, the (empirically) worse natural language parser achieved the better score on our artificial data. URNNG also provides the best language models across configurations, which is less surprising (perplexity scores estimated on training data are available in the appendix). Both models show a tendency to induce linear structures in the configuration `-sr-mce`, especially when less data is available. However, they seem to have opposite biases with CPCFG preferring **left-branching** structures, and URNNG **right-branching** ones.

Differences in `mce` impact `mce` strongly impacts all three models, but the impact on CPCFG is particularly dramatic, as no model ever beats the right-branching baseline in any of the `-mce` configuration. We find it remarkable, that despite plausibly able to express `mce` without branching structure (see § 4.2), the `-sr+mce` configuration pushes URNNG towards inducing NP constituents (though, not CP constituents).

Differences in `sr` impact The effect of selectional restrictions alone (`+sr-mce`) varies across models: CPCFG and NPCFG improve NP induction but not CPs, whereas URNNG shows the opposite pattern, especially excelling at CP recall with 100k data. Interestingly, NPCFG's F1 score drops sharply under `-sr`, though NP recall remains relatively stable. The contrast between `+sr+mce` and `-sr+mce` is more pronounced in NPCFG and URNNG than in CPCFG, probably because CPCFG cannot explicitly model selectional restrictions with its context vector z. Since z uniformely affects every occurence of a nonterminal symbol, lexical items generated from the same symbol share the same distribution, and CPCFG must assign different symbols to nouns with different semantic features. Consequently it has to encode semantic variation within a fixed inventory of 30 nonterminals and 60 preterminals. In contrast, URNNG may exploit its hidden state to model such distinctions more flexibly.

Performance correlations We used Spearman's rank correlation to assess the relationship between perplexity, training size, and syntactic performance (F1, NP/CP/VP recall; see Appendix C). CPCFG

and URNNG showed strong negative correlations between perplexity and performance, especially NP recall, suggesting that lower perplexity often aligns with better parsing. This trend was less clear for NPCFG. In contrast, correlations with training size were weaker. With small datasets, models typically saw the full data multiple times before converging, so seed-related variation mainly reflected data ordering. For larger datasets, perplexity often plateaued early, leading to convergence before full data exposure and greater sensitivity to the specific subset encountered.

Robustness to semantic and syntactic variation We further evaluated our models using test sets from the `-sr-mce` configuration and, for the models trained on `+mce`, `-sr+mce` configurations. Parsing performance remained consistent across these settings, suggesting that models do not rely on semantic cues, and that (when exposed to both kind of RCs) they learn to treat subject and object RC similarly. This indicates that self-embedding structures are either jointly acquired or jointly missed.

6 Limitations

The tested latent tree models obviously have very high variance under most configurations (URNNG on `+sr+mce` being an exception). Though the problem is pervasive in grammar induction, additional runs could help increase statistical significance. Second, comparison with more models would be very informative. In particular a comparison between the recent Tensor Decomposition PCFG model (Yang et al., 2021), since the former increased number of symbols could maybe overcome CPCFG apparently limitation to benefit from `sr`. Another limitation lies in the latent non/preterminal symbols in CPCFG and NPCFG, which we did not analyze in detail; future work is needed to better understand how these symbols relate to syntactic and semantic categories. Finally verbs and nouns are very unbalanced in our lexicon (more than in reality) and this asymmetry could have some effects, *e.g.* on some models' preferences for a given flow of information (from subject to verb vs. from verb to subject).

7 Conclusion

We have designed a controlled experimental setting to assess the respective effect of two linguistic phenomena (one categorical, multiple center embedding and one semantic, selectional restrictions) on latent tree models induction. Testing three well established latent tree baselines in these settings allows to make general observations on the relative strength of the two phenomena, and report differences in their impact on the tested models. While we focused on constituency models in this study, our methodology and data are readily applicable to the dependency setting, and testing dependency latent models is one of our future avenues of research.

Acknowledgments

We would like to thank anonymous reviewers for their suggestions and comments. This research was funded by the Natural Sciences and Engineering Research Council of Canada (RN001462).

References

C.L. Baker. 1995. *English Syntax*. MIT Press.

Yonatan Bisk and Julia Hockenmaier. 2013. An HDP model for inducing Combinatory Categorial Grammars. *Transactions of the Association for Computational Linguistics*, 1:75–88.

Glenn Carroll and Eugene Charniak. 1992. *Two experiments on learning probabilistic dependency grammars from corpora*. Department of Computer Science, Univ.

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

Morten H. Christiansen and Maryellen C. MacDonald. 2009. A usage-based approach to recursion in sentence processing. *Language Learning*, 59:126–161.

Alexander Clark. 2001. Unsupervised induction of stochastic context-free grammars using distributional clustering. In *Proceedings of the ACL 2001 Workshop on Computational Natural Language Learning (ConLL)*.

Jason Eisner and Giorgio Satta. 1999. Efficient parsing for bilexical context-free grammars and head automaton grammars. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics*, pages 457–464, College Park, Maryland, USA. Association for Computational Linguistics.

Phu Mon Htut, Kyunghyun Cho, and Samuel Bowman. 2018. Grammar induction with neural language models: An unusual replication. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4998–5003, Brussels, Belgium. Association for Computational Linguistics.

Lifeng Jin, Finale Doshi-Velez, Timothy Miller, William Schuler, and Lane Schwartz. 2018. Unsupervised grammar induction with depth-bounded PCFG. *Transactions of the Association for Computational Linguistics*, 6:211–224.

Fred Karlson. 2007. Constraints on multiple center-embedding of clauses. *Journal of Linguistics*, 43(2):365–392.

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.

Dan Klein and Christopher Manning. 2004. Corpus-based induction of syntactic structure: Models of dependency and constituency. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL-04)*, pages 478–485, Barcelona, Spain.

Dan Klein and Christopher D Manning. 2001. Distributional phrase structure induction. In *Proceedings of the ACL 2001 Workshop on Computational Natural Language Learning (ConLL)*.

Dan Klein and Christopher D. Manning. 2002. A generative constituent-context model for improved grammar induction. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 128–135, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Adhiguna Kuncoro, Miguel Ballesteros, Lingpeng Kong, Chris Dyer, Graham Neubig, and Noah A Smith. 2017. What do recurrent neural network grammars learn about syntax? In *EACL (1)*.

Nur Lan, Michal Geyer, Emmanuel Chemla, and Roni Katzir. 2022. Minimum description length recurrent neural networks. *Transactions of the Association for Computational Linguistics*, 10:785–799.

Guy Lapalme. 2020. The jsrealb text realizer: Organization and use cases. *arXiv preprint arXiv:2012.15425*.

K. Lari and S. J. Young. 1990. The estimation of stochastic context-free grammars using the inside-outside algorithm. *Computer Speech and Language*, 4:35–56.

Bingzhi Li, Guillaume Wisniewski, and Benoît Crabbé. 2023. Assessing the capacity of transformer to abstract syntactic representations: A contrastive analysis based on long-distance agreement. *Transactions of the Association for Computational Linguistics*, 11:18–33.

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.

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.

Louis Martin, Benjamin Muller, Pedro Javier Ortiz Suárez, Yoann Dupont, Laurent Romary, Éric de la Clergerie, Djamé Seddah, and Benoît Sagot. 2020. CamemBERT: a tasty French language model. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7203–7219, Online. Association for Computational Linguistics.

Rebecca Marvin and Tal Linzen. 2018. Targeted syntactic evaluation of language models. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1192–1202, Brussels, Belgium. Association for Computational Linguistics.

William Merrill, Gail Weiss, Yoav Goldberg, Roy Schwartz, Noah A. Smith, and Eran Yahav. 2020. A formal hierarchy of RNN architectures. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 443–459, Online. Association for Computational Linguistics.

Paul Molins and Guy Lapalme. 2015. Jsrealb: A bilingual text realizer for web programming. In *Proceedings of the 15th European Workshop on Natural Language Generation (ENLG)*, pages 109–111.

Mark-Jan Nederhof and Giorgio Satta. 2011. Splittability of bilexical context-free grammars is undecidable. *Computational Linguistics*, 37(4):867–879.

Barbara H. Partee, Alice ter Meulen, and Robert E. Wall. 1990. *Mathematical Methods in Linguistics. Corrected first edition.* Kluwer Academic Publishers, Dordrecht.

John K Pate and Mark Johnson. 2016. Grammar induction from (lots of) words alone. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 23–32, Osaka, Japan. The COLING 2016 Organizing Committee.

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

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*.

C. Tellier. 2003. *Éléments de syntaxe du français: méthodes d'analyse en grammaire générative*. Morin.

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.

Songlin Yang, Yanpeng Zhao, and Kewei Tu. 2021. PCFGs can do better: Inducing probabilistic context-free grammars with many symbols. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1487–1498, Online. Association for Computational Linguistics.

Shunyu Yao, Binghui Peng, Christos Papadimitriou, and Karthik Narasimhan. 2021. Self-attention networks can process bounded hierarchical languages. 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 3770–3785, Online. Association for Computational Linguistics.

Hao Zhu, Yonatan Bisk, and Graham Neubig. 2020. The return of lexical dependencies: Neural lexicalized pcfgs. *Transactions of the Association for Computational Linguistics*, 8:647–661.

A Hyperparameters

The hyperparameters used in our experiments largely follow those reported in prior work (Kim et al., 2019a,b). We conducted four runs for each model (CPCFG, NPCFG, and URNNG) with the following randomly selected seeds: `3435`, `648708704`, `1320159950`, and `603135965`. Table 2 summarizes the hyperparameter settings. In the experiments with the NPCFG, the `z_dim` parameter of the CPCFG was set to 0.

Table 2: Hyperparameters

CPCFG (NPCFG)

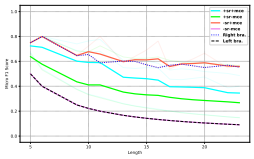
Description	Value	Flag
latent dimension	64	`-z_dim`
number of preterminal states	60	`-t_states`
number of nonterminal states	30	`-nt_states`
symbol embedding dimension	256	`-state_dim`
hidden dim for variational LSTM	512	`-h_dim`
embedding dim for variational LSTM	512	`-w_dim`
starting learning rate	0.001	`-lr`
gradient clipping	3	`-max_grad_norm`
max sentence length cutoff start	30	`-max_length`
increment max length each epoch	1	`-len_incr`
final max length cutoff	40	`-final_max_length`
Adam β_1	0.75	`-beta1`
Adam β_2	0.999	`-beta2`
which GPU to use	0	`-gpu`
validation every N steps	3000	`-val_every`
increment max length every N steps	3000	`-incr_step`
early stopping patience (epochs)	5	`-early_stopping_patience`
minimum training steps	10000	`-min_steps`

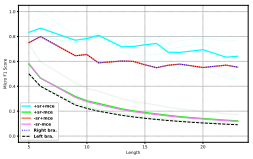
URNNG

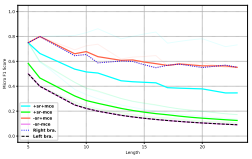
Description	Value	Flag
hidden dim (LM/RNNG)	650	`-w_dim`
hidden dim (LM/RNNG)	650	`-h_dim`
hidden dim (variational RNN)	256	`-q_dim`
number of layers (LM & stack LSTM)	2	`-num_layers`
dropout rate	0.5	`-dropout`
include EOS in val PPL (0/1)	0	`-count_eos_ppl`
no LR decay before this	8	`-min_epochs`
IWAE samples (eval)	5	`-mc_samples`
samples for score-function grads	8	`-samples`
starting learning rate	1	`-lr`
LR for inference network q	0.0001	`-q_lr`
LR for action layer	0.1	`-action_lr`
LR decay factor	0.5	`-decay`
KL warmup steps	10000	`-kl_warmup`
steps to train q	10000	`-train_q_steps`
uniform init range	0.1	`-param_init`
grad clipping (model)	5	`-max_grad_norm`
grad clipping (q)	1	`-q_max_grad_norm`
validation every N steps	3000	`-val_every`
minimum training steps	1500	`-min_steps`

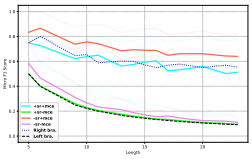
B All scores by length and All Results tables

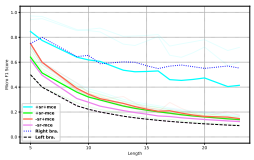
F1 score

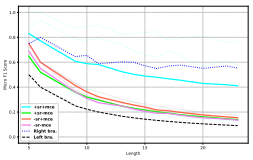

CPCFG F1 3k

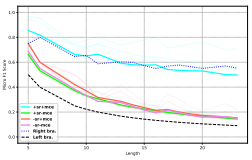

CPCFG F1 12k

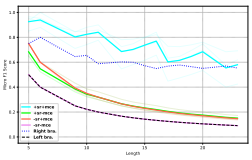

CPCFG F1 100k

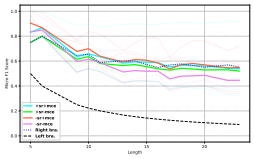

CPCFG F1 400k

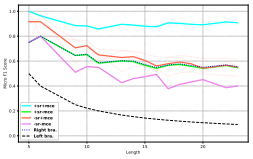

NPCFG F1 3k

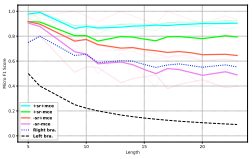

NPCFG F1 12k

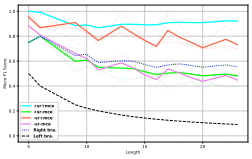

NPCFG F1 100k


NPCFG F1 400k

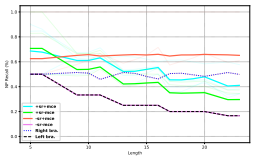

URNNG F1 3k

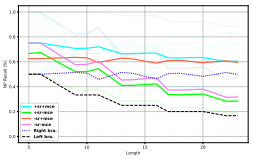

URNNG F1 12k

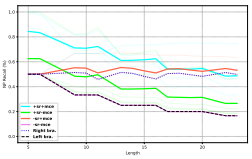

URNNG F1 100k

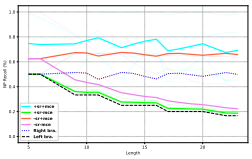

URNNG F1 400k

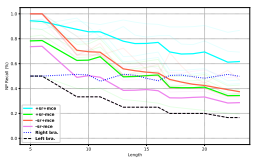
NP prediction recall

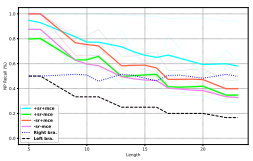

CPCFG NP 3k

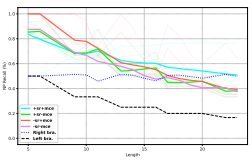

CPCFG NP 12k

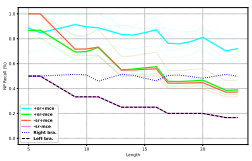

CPCFG NP 100k

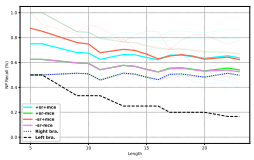

CPCFG NP 400k

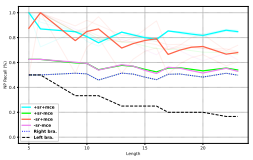

NPCFG NP 3k

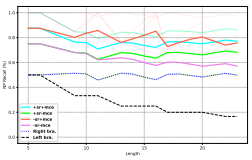

NPCFG NP 12k

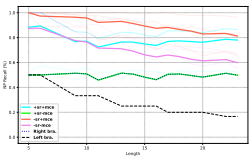

NPCFG NP 100k


NPCFG NP 400k

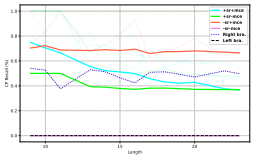

URNNG NP 3k


URNNG NP 12k

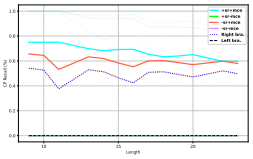

URNNG NP 100k


URNNG NP 400k

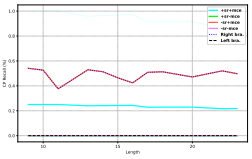
CP prediction recall



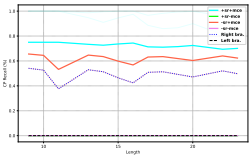
CPCFG CP 3k



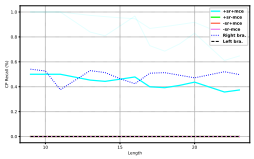
CPCFG CP 12k



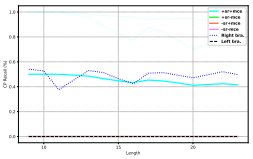
CPCFG CP 100k



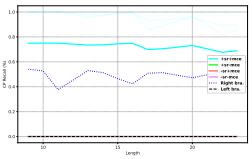
CPCFG CP 400k



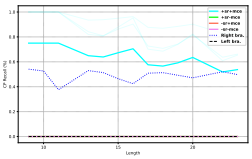
NPCFG CP 3k



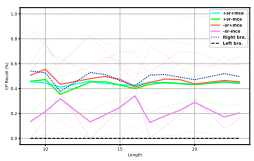
NPCFG CP 12k



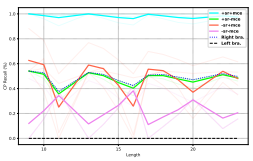
NPCFG CP 100k



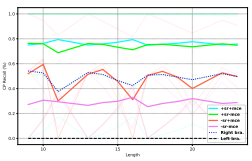
NPCFG CP 400k



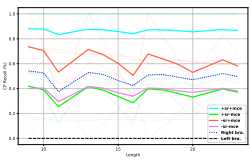
URNNG CP 3k



URNNG CP 12k

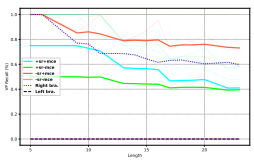


URNNG CP 100k

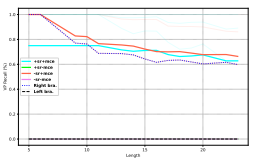


URNNG CP 400k

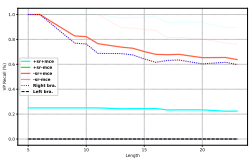
VP prediction recall



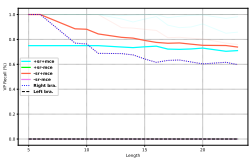
CPCFG VP 3k



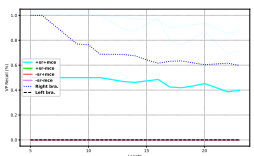
CPCFG VP 12k



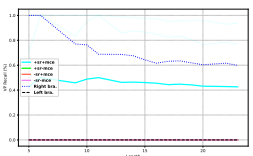
CPCFG VP 100k



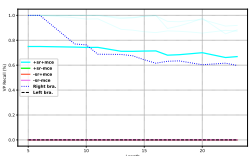
CPCFG VP 400k



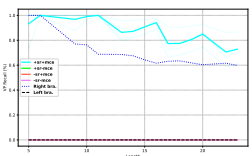
NPCFG VP 3k



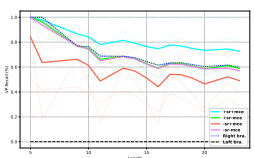
NPCFG VP 12k



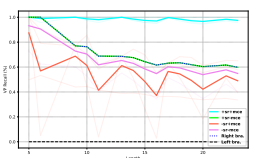
NPCFG VP 100k



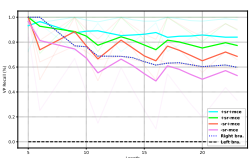
NPCFG VP 400k



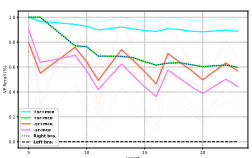
URNNG VP 3k



URNNG VP 12k



URNNG VP 100k



URNNG VP 400k

Table 3: CPCFG Results

Data Type	Size	Seed	Val PPL	Last Step	**F1**	**NP**	**CP**	**VP**	**nb NT**	**nb PreT**
+sr+mce	3k	0	24.93	9000	0.51	0.42	0.58	0.65	10	30
		1	24.41	10000	0.58	0.6	0.66	0.72	10	28
		2	23.43	11000	0.23	0.52	0.0	0.0	10	33
		3	22.03	9000	0.54	0.48	0.63	0.75	14	32
	12k	0	20.63	13000	0.73	0.71	0.92	0.95	16	30
		1	18.97	12000	0.61	0.66	0.7	0.8	9	22
		2	20.18	9500	0.77	0.75	1.0	1.0	10	23
		3	21.74	8000	0.22	0.49	0.0	0.0	11	33
	100k	0	20.11	11500	0.78	0.72	0.92	0.95	12	28
		1	22.14	10500	0.26	0.63	0.0	0.0	11	25
		2	22.13	8000	0.22	0.49	0.0	0.0	11	30
		3	20.43	12500	0.24	0.54	0.0	0.0	7	24
	400k	0	20.17	13500	0.67	0.54	0.89	0.93	11	32
		1	18.87	11000	0.77	0.75	1.0	1.0	11	25
		2	19.29	13000	0.26	0.62	0.0	0.0	10	28
		3	18.5	14000	0.84	0.99	0.99	0.99	8	29
+sr-mce	3k	0	18.54	8500	0.62	0.39	1.0	1.0	13	28
		1	21.56	7000	0.15	0.25	0.0	0.0	7	21
		2	19.4	9000	0.23	0.51	0.0	0.0	6	26
		3	18.34	9000	0.52	0.5	0.56	0.73	7	30
	12k	0	19.79	11000	0.16	0.27	0.0	0.0	8	32
		1	16.6	15500	0.26	0.63	0.0	0.0	8	34
		2	19.36	10000	0.16	0.28	0.0	0.0	7	29
		3	18.64	10500	0.2	0.41	0.0	0.0	7	25
	100k	0	19.71	8500	0.17	0.3	0.0	0.0	6	24
		1	16.43	17500	0.27	0.64	0.0	0.0	7	38
		2	19.38	10500	0.15	0.27	0.0	0.0	7	29
		3	20.05	9500	0.15	0.26	0.0	0.0	8	25
	400k	0	19.17	7000	0.15	0.25	0.0	0.0	7	22
		1	20.01	6500	0.15	0.26	0.0	0.0	7	22
		2	19.22	10000	0.16	0.29	0.0	0.0	7	30
		3	19.75	8500	0.16	0.28	0.0	0.0	6	26
-sr+mce	3k	0	39.23	10000	0.64	0.61	0.72	0.81	13	29
		1	47.51	7000	0.59	0.5	0.49	0.66	7	22
		2	46.74	6500	0.59	0.5	0.49	0.66	7	18
		3	34.58	11500	0.85	1.0	1.0	1.0	9	23
	12k	0	43.03	8000	0.59	0.5	0.49	0.66	7	16
		1	45.7	6500	0.59	0.5	0.49	0.66	6	16
		2	44.68	6500	0.59	0.5	0.49	0.66	6	15
		3	35.25	10000	0.79	0.94	0.89	0.93	7	14
	100k	0	44.93	8000	0.59	0.5	0.49	0.66	7	14
		1	41.64	7000	0.62	0.64	0.49	0.85	11	12
		2	44.21	8000	0.59	0.5	0.49	0.66	8	16
		3	44.6	7500	0.59	0.5	0.49	0.66	6	12
	400k	0	34.49	14000	0.62	0.64	0.49	0.85	12	25
		1	32.42	12000	0.85	1.0	1.0	1.0	9	20
		2	44.95	7000	0.59	0.5	0.49	0.66	7	12
		3	44.75	8000	0.59	0.5	0.49	0.66	8	18
-sr-mce	3k	0	42.17	6500	0.15	0.25	0.0	0.0	6	19
		1	37.25	7000	0.15	0.25	0.0	0.0	7	22
		2	42.7	6500	0.15	0.25	0.0	0.0	6	16
		3	42.31	6500	0.15	0.25	0.0	0.0	6	18
	12k	0	28.14	13000	0.27	0.64	0.0	0.0	7	17
		1	39.5	7000	0.15	0.25	0.0	0.0	6	23
		2	38.91	6500	0.15	0.25	0.0	0.0	6	18
		3	32.06	14500	0.27	0.64	0.0	0.0	7	19
	100k	0	38.56	8000	0.15	0.25	0.0	0.0	6	27
		1	38.83	6500	0.15	0.25	0.0	0.0	6	22
		2	38.87	7000	0.15	0.25	0.0	0.0	6	23
		3	34.74	6500	0.15	0.25	0.0	0.0	6	18
	400k	0	38.56	7500	0.15	0.25	0.0	0.0	6	25
		1	38.72	8000	0.15	0.25	0.0	0.0	7	21
		2	32.36	9500	0.24	0.56	0.0	0.0	7	21
		3	34.44	8000	0.15	0.25	0.0	0.0	7	22

Table 4: NPCFG Results

Data Type	Size	Seed	Val PPL	Last Step	**F1**	**NP**	**CP**	**VP**	**nb NT**	**nb PreT**
+sr+mce	3k	0	20.67	9000	0.72	0.81	0.77	0.85	9	28
		1	25.58	6500	0.25	0.6	0.0	0.0	9	23
		2	25.48	6500	0.27	0.64	0.0	0.0	7	25
		3	23.28	8500	0.78	0.9	0.9	0.94	11	25
	12k	0	25.98	7000	0.2	0.42	0.0	0.0	7	22
		1	21.91	10000	0.82	0.97	0.95	0.97	7	23
		2	24.42	7500	0.27	0.64	0.0	0.0	7	29
		3	21.01	10500	0.67	0.69	0.82	0.83	10	24
	100k	0	22.71	10000	0.27	0.64	0.0	0.0	8	23
		1	20.36	9000	0.62	0.48	1.0	0.89	12	22
		2	20.51	11500	0.64	0.55	0.92	0.94	10	18
		3	25.2	8000	0.75	0.73	0.95	0.96	8	22
	400k	0	21.74	12500	0.69	0.71	0.77	0.85	10	34
		1	20.65	9000	0.77	0.87	0.9	0.93	11	26
		2	21.19	9500	0.73	0.88	0.77	0.81	10	20
		3	20.25	14000	0.56	0.76	0.0	0.77	14	33
+sr-mce	3k	0	22.32	9500	0.22	0.5	0.0	0.0	8	26
		1	19.56	9500	0.25	0.58	0.0	0.0	8	28
		2	19.99	8500	0.24	0.54	0.0	0.0	8	23
		3	22.52	6500	0.16	0.29	0.0	0.0	7	28
	12k	0	21.5	8500	0.21	0.47	0.0	0.0	8	27
		1	18.72	9500	0.22	0.49	0.0	0.0	8	24
		2	18.6	12000	0.22	0.49	0.0	0.0	12	27
		3	18.95	10000	0.22	0.49	0.0	0.0	12	24
	100k	0	18.65	10500	0.22	0.49	0.0	0.0	8	26
		1	18.81	8500	0.24	0.54	0.0	0.0	8	31
		2	18.84	12000	0.22	0.49	0.0	0.0	11	24
		3	21.26	11000	0.25	0.58	0.0	0.0	8	26
	400k	0	18.83	7500	0.22	0.49	0.0	0.0	8	21
		1	18.63	12000	0.22	0.48	0.0	0.0	9	31
		2	18.72	11000	0.27	0.64	0.0	0.0	8	22
		3	19.45	8500	0.23	0.53	0.0	0.0	8	24
-sr+mce	3k	0	36.07	9500	0.22	0.49	0.0	0.0	8	17
		1	31.45	10000	0.23	0.52	0.0	0.0	8	16
		2	38.44	8000	0.26	0.61	0.0	0.0	12	20
		3	36.33	10000	0.22	0.5	0.0	0.0	6	16
	12k	0	43.74	8000	0.27	0.64	0.0	0.0	6	13
		1	35.22	10000	0.22	0.49	0.0	0.0	6	17
		2	48.96	7500	0.24	0.57	0.0	0.0	6	18
		3	34.2	9500	0.24	0.56	0.0	0.0	9	18
	100k	0	35.9	10500	0.27	0.66	0.0	0.0	13	13
		1	35.26	10000	0.22	0.5	0.0	0.0	8	16
		2	43.02	8000	0.23	0.53	0.0	0.0	8	15
		3	40.46	7000	0.24	0.57	0.0	0.0	8	19
	400k	0	40.92	8000	0.27	0.64	0.0	0.0	7	12
		1	35.78	8000	0.22	0.49	0.0	0.0	7	16
		2	43.69	7000	0.22	0.49	0.0	0.0	9	16
		3	33.54	11000	0.23	0.52	0.0	0.0	8	14
-sr-mce	3k	0	27.2	8000	0.25	0.59	0.0	0.0	10	20
		1	39.07	7000	0.15	0.25	0.0	0.0	6	19
		2	31.22	10000	0.21	0.46	0.0	0.0	8	14
		3	35.33	6500	0.15	0.25	0.0	0.0	5	10
	12k	0	27.04	8000	0.26	0.63	0.0	0.0	7	15
		1	32.63	8000	0.22	0.49	0.0	0.0	11	15
		2	34.11	6500	0.15	0.25	0.0	0.0	5	14
		3	26.57	9000	0.22	0.49	0.0	0.0	10	16
	100k	0	27.12	10000	0.31	0.79	0.0	0.0	8	13
		1	33.06	6500	0.23	0.54	0.0	0.0	7	13
		2	34.46	6500	0.15	0.25	0.0	0.0	8	12
		3	26.79	7500	0.22	0.49	0.0	0.0	9	15
	400k	0	33.5	7000	0.15	0.25	0.0	0.0	6	18
		1	33.13	7500	0.15	0.25	0.0	0.0	6	19
		2	33.73	8000	0.15	0.25	0.0	0.0	6	25
		3	33.98	8000	0.15	0.25	0.0	0.0	7	12

Table 5: URNNG Results

Data Type	Size	Seed	Val PPL	Last Step	**F1**	**NP**	**CP**	**VP**
+sr+mce	3k	0	15.96	3500	0.43	0.5	0.0	0.66
		1	14.25	4500	0.91	0.85	0.99	0.99
		2	16.24	3500	0.42	0.47	0.01	0.63
		3	14.53	4500	0.79	0.81	0.77	0.82
	12k	0	11.78	8500	0.91	0.84	1.0	1.0
		1	12.45	6000	0.9	0.84	0.99	0.97
		2	12.64	6000	0.89	0.83	0.95	0.97
		3	11.84	8500	0.89	0.82	0.98	0.99
	100k	0	12.54	6000	0.87	0.85	0.99	0.82
		1	11.69	8000	0.91	0.85	1.0	0.98
		2	11.47	9000	0.91	0.85	1.0	0.98
		3	12.48	7000	0.44	0.5	0.06	0.66
	400k	0	11.71	7500	0.91	0.85	1.0	0.98
		1	11.57	8500	0.91	0.85	1.0	0.98
		2	11.87	7500	0.91	0.85	1.0	0.98
		3	12.2	8500	0.52	0.53	0.46	0.67
+sr-mce	3k	0	13.27	5500	0.59	0.5	0.49	0.66
		1	13.32	4500	0.51	0.5	0.27	0.66
		2	13.58	3500	0.58	0.5	0.49	0.66
		3	12.86	5500	0.65	0.74	0.49	0.62
	12k	0	11.59	6000	0.57	0.5	0.45	0.66
		1	11.34	6000	0.58	0.5	0.49	0.66
		2	11.75	5500	0.58	0.5	0.49	0.66
		3	10.88	6500	0.66	0.74	0.49	0.66
	100k	0	10.28	9000	0.91	0.85	1.0	1.0
		1	10.71	8000	0.59	0.5	0.49	0.66
		2	10.37	8000	0.89	0.85	1.0	0.89
		3	10.91	7000	0.59	0.5	0.49	0.66
	400k	0	10.84	7500	0.59	0.5	0.49	0.66
		1	11.09	7000	0.46	0.5	0.15	0.66
		2	11.53	6000	0.53	0.5	0.34	0.66
		3	10.83	7500	0.59	0.5	0.49	0.66
-sr+mce	3k	0	42.06	3500	0.43	0.49	0.0	0.64
		1	42.11	4000	0.75	0.82	0.63	0.74
		2	39.79	4500	0.62	0.75	0.59	0.42
		3	50.4	4500	0.56	0.66	0.63	0.34
	12k	0	37.71	5000	0.67	0.76	0.52	0.63
		1	35.82	6000	0.54	0.63	0.34	0.68
		2	35.84	6000	0.62	0.81	0.42	0.42
		3	35.61	6000	0.63	0.79	0.66	0.37
	100k	0	35.52	7000	0.95	0.98	0.92	0.92
		1	35.54	7000	0.67	0.82	0.4	0.64
		2	35.37	7000	0.46	0.5	0.16	0.66
		3	35.45	6500	0.64	0.86	0.45	0.7
	400k	0	35.38	8000	0.84	0.99	0.72	0.69
		1	35.32	9000	0.76	0.91	0.63	0.61
		2	35.36	8000	0.76	0.89	0.64	0.63
		3	35.42	7500	0.63	0.75	0.48	0.52
-sr-mce	3k	0	44.85	3500	0.46	0.5	0.16	0.66
		1	35.63	3500	0.43	0.48	0.03	0.64
		2	44.61	3500	0.65	0.74	0.49	0.61
		3	36.76	5500	0.46	0.5	0.16	0.66
	12k	0	33.37	6500	0.46	0.5	0.16	0.66
		1	43.12	5000	0.46	0.5	0.16	0.66
		2	33.41	6000	0.46	0.5	0.16	0.66
		3	36.63	5000	0.58	0.73	0.37	0.46
	100k	0	33.33	6000	0.46	0.5	0.15	0.66
		1	43.01	5000	0.66	0.74	0.49	0.66
		2	35.19	5000	0.54	0.72	0.36	0.38
		3	33.2	7500	0.46	0.5	0.16	0.66
	400k	0	35.73	5000	0.46	0.5	0.16	0.66
		1	33.25	8500	0.57	0.78	0.55	0.28
		2	33.21	7500	0.53	0.68	0.34	0.41
		3	38.0	6500	0.66	0.74	0.49	0.66

C Spearman's Test Results

Table 6: Spearman's Test Results: CPCFG

Data Type	Pair	Spearman's ρ	p-value	Significance
+sr+mce	PPL-Size	-0.7276	0.0014	**
	F1-Size	0.2795	0.2944	n.s.
	F1-PPL	-0.6013	0.0137	*
	NP-Size	0.4983	0.0495	*
	NP-PPL	-0.7119	0.002	**
	CP-Size	0.1933	0.4732	n.s.
	CP-PPL	-0.5535	0.0261	*
	VP-Size	0.1933	0.4732	n.s.
	VP-PPL	-0.5656	0.0224	*
+sr-mce	PPL-Size	0.1576	0.5598	n.s.
	F1-Size	-0.5209	0.0385	*
	F1-PPL	-0.728	0.0014	**
	NP-Size	-0.3588	0.1723	n.s.
	NP-PPL	-0.733	0.0012	**
	CP-Size	-0.506	0.0455	*
	CP-PPL	-0.4065	0.1182	n.s.
	VP-Size	-0.506	0.0455	*
	VP-PPL	-0.4065	0.1182	n.s.
-sr+mce	PPL-Size	-0.2183	0.4167	n.s.
	F1-Size	-0.0489	0.8574	n.s.
	F1-PPL	-0.8415	0.0	***
	NP-Size	0.0209	0.9386	n.s.
	NP-PPL	-0.8483	0.0	***
	CP-Size	-0.2153	0.4231	n.s.
	CP-PPL	-0.6712	0.0044	**
	VP-Size	0.0209	0.9386	n.s.
	VP-PPL	-0.8483	0.0	***
-sr-mce	PPL-Size	-0.4672	0.068	n.s.
	F1-Size	0.0356	0.8958	n.s.
	F1-PPL	-0.6811	0.0037	**
	NP-Size	0.0356	0.8958	n.s.
	NP-PPL	-0.6811	0.0037	**
	CP-Size	NaN	NaN	n.s.
	CP-PPL	NaN	NaN	n.s.
	VP-Size	NaN	NaN	n.s.
	VP-PPL	NaN	NaN	n.s.

Table 7: Spearman's Test Results: NPCFG

Data Type	Pair	Spearman's ρ	p-value	Significance
+sr+mce	PPL-Size	-0.5457	0.0288	*
	F1-Size	0.1946	0.4702	n.s.
	F1-PPL	-0.3392	0.1987	n.s.
	NP-Size	0.1216	0.6536	n.s.
	NP-PPL	-0.2094	0.4363	n.s.
	CP-Size	0.1814	0.5014	n.s.
	CP-PPL	-0.4111	0.1137	n.s.
	VP-Size	0.1788	0.5077	n.s.
	VP-PPL	-0.3902	0.1351	n.s.
+sr-mce	PPL-Size	-0.5461	0.0286	*
	F1-Size	0.2271	0.3977	n.s.
	F1-PPL	-0.111	0.6823	n.s.
	NP-Size	0.0873	0.7479	n.s.
	NP-PPL	0.031	0.9092	n.s.
	CP-Size	NaN	NaN	n.s.
	CP-PPL	NaN	NaN	n.s.
	VP-Size	NaN	NaN	n.s.
	VP-PPL	NaN	NaN	n.s.
-sr+mce	PPL-Size	0.097	0.7208	n.s.
	F1-Size	0.0189	0.9447	n.s.
	F1-PPL	0.314	0.2363	n.s.
	NP-Size	-0.0061	0.982	n.s.
	NP-PPL	0.3403	0.1972	n.s.
	CP-Size	NaN	NaN	n.s.
	CP-PPL	NaN	NaN	n.s.
	VP-Size	NaN	NaN	n.s.
	VP-PPL	NaN	NaN	n.s.
-sr-mce	PPL-Size	0.0243	0.929	n.s.
	F1-Size	-0.2534	0.3436	n.s.
	F1-PPL	-0.8164	0.0001	***
	NP-Size	-0.2534	0.3436	n.s.
	NP-PPL	-0.8164	0.0001	***
	CP-Size	NaN	NaN	n.s.
	CP-PPL	NaN	NaN	n.s.
	VP-Size	NaN	NaN	n.s.
	VP-PPL	NaN	NaN	n.s.

Table 8: Spearman's Test Results: URNNG

Data Type	Pair	Spearman's ρ	p-value	Significance
+sr+mce	PPL-Size	-0.7155	0.0018	**
	F1-Size	0.4054	0.1193	n.s.
	F1-PPL	-0.7405	0.001	**
	NP-Size	0.4818	0.0588	n.s.
	NP-PPL	-0.6258	0.0095	**
	CP-Size	0.5371	0.0319	*
	CP-PPL	-0.8149	0.0001	***
	VP-Size	0.1543	0.5682	n.s.
	VP-PPL	-0.6229	0.01	**
+sr-mce	PPL-Size	-0.7155	0.0018	**
	F1-Size	0.0	1.0	n.s.
	F1-PPL	-0.5779	0.019	*
	NP-Size	-0.0799	0.7688	n.s.
	NP-PPL	-0.4339	0.0931	n.s.
	CP-Size	-0.0349	0.898	n.s.
	CP-PPL	-0.5142	0.0416	*
	VP-Size	0.3202	0.2266	n.s.
	VP-PPL	-0.5911	0.0159	*
-sr+mce	PPL-Size	-0.9459	0.0	***
	F1-Size	0.5595	0.0242	*
	F1-PPL	-0.4381	0.0897	n.s.
	NP-Size	0.5769	0.0193	*
	NP-PPL	-0.4875	0.0554	n.s.
	CP-Size	0.2433	0.364	n.s.
	CP-PPL	-0.174	0.5192	n.s.
	VP-Size	0.2005	0.4565	n.s.
	VP-PPL	-0.1975	0.4635	n.s.
-sr-mce	PPL-Size	-0.4972	0.0501	n.s.
	F1-Size	0.3824	0.1438	n.s.
	F1-PPL	0.2374	0.376	n.s.
	NP-Size	0.4289	0.0974	n.s.
	NP-PPL	0.1261	0.6417	n.s.
	CP-Size	0.381	0.1454	n.s.
	CP-PPL	0.1756	0.5154	n.s.
	VP-Size	-0.1602	0.5533	n.s.
	VP-PPL	0.2585	0.3336	n.s.

D Statistics of Non-/Pre-terminal Symbols for CPCFG and NPCFG

Table 9: Mean number of Non-/Pre-terminal symbols

Model	Data Type	Mean nb NT	(std)	Mean nb PreT	(std)
CPCFG	+sr+mce	10.69	(2.05)	28.25	(3.44)
	+sr-mce	7.38	(1.58)	27.56	(4.49)
	-sr+mce	8.12	(2.09)	17.62	(4.83)
	-sr-mce	6.38	(0.48)	20.69	(2.93)
NPCFG	+sr+mce	9.38	(1.96)	24.81	(4.22)
	+sr-mce	8.69	(1.49)	25.75	(2.8)
	-sr+mce	8.06	(1.95)	16.0	(2.15)
	-sr-mce	7.44	(1.77)	15.62	(3.66)