Inside-Outside Algorithm for Probabilistic Product-Free Lambek Categorial Grammar

Jinman Zhao and Gerald Penn
Dept. of Computer Science
University of Toronto
CANADA
{jzhao,gpenn}@cs.toronto.edu

Abstract

The inside-outside algorithm is widely utilized in statistical models related to context-free grammars. It plays a key role in the EM estimation of probabilistic context-free grammars. In this work, we introduce an inside-outside algorithm for Probabilistic Lambek Categorical Grammar (PLCG).[1]

1 Introduction

Many studies have discovered hidden syntactic structures within language models (Shen et al., 2018; Niu et al., 2022; Millière, 2024). These findings suggest that, even though (large) language models are primarily trained on sequential text data without the guidance of explicit rules, they learn some aspects of an inherent grammar. Evidence for such latent acquisition reinforces the idea that grammar plays a critical role in language comprehension and generation, even within complex machine learning models.

Lambek Categorial Grammar (LCG) bridges syntax and semantics through a transparent, compositional system of labeled-term deduction grounded in the lambda calculus. LCG lexicalizes grammar to such an extent that it relies on only four fixed grammar schemata, making it a compact yet powerful tool for syntactic and semantic analysis. Recent advancements in quantum NLP, as discussed in Wu et al. (2021), have advocated for the adoption of pregroup grammar (a variant of LCG) for mapping linguistic structures into quantum systems.

Parsing plays a key role in the study of grammar since it provides deep insights into language understanding (Shayegh et al., 2024). Parsing shows promising applications in multilingual and multimodal environments (Amara et al., 2024; Kanayama et al., 2024). Parsing and probabilistic inference in computational linguistics often rely on algorithms that efficiently compute probabilities associated with various structures. One of the most renowned examples of such algorithms is the Inside-Outside (Lari and Young, 1990) algorithm, originally developed for Probabilistic Context-Free Grammars (PCFGs) (Huang and Fu, 1971). The inside-outside algorithm is arguably *the* standard parameter re-estimation method for context-free grammars, being the PCFG realization of expectation-maximization.

Zhao and Penn (2021) formulated the first generative probabilistic model for LCG. Zhao and Penn (2024) propose a better generative model for Probabilistic Lambek Categorial Grammar (PLCG) which translates the LCG sequent into a labelled tree structure.

In this paper, we augment that work with a novel Inside-Outside algorithm specifically designed for PLCG. Below are our main contributions:

- a modified generative model based upon Zhao and Penn (2024) that is suitable for the purposes of inside-outside;
- an inside-outside algorithm for PLCG; and
- an empirical evaluation of the algorithm on LCGbank (Bhargava et al., 2024), showing that our algorithm can significantly boost the average probability of this corpus.

2 Preliminaries

2.1 Inside-Outside Algorithm for PCFG

The inside-outside Algorithm is an EM-based algorithm for PCFGs. The inputs are a PCFG $\mathcal{G}$, which can be randomly generated or learned from an annotated corpus, and a new corpus of sentences.

2.1.1 Probabilistic Context-Free Grammar

A context-free grammar in Chomsky Normal Form (CNF) can be represented in the 4-tuple form. $\mathcal{G} = (N, \Sigma, R, S)$ where:

[1] https://github.com/zhaojinm/Inside-outside-plcg

- N is a finite set that contains nonterminal variables,
- Σ is a finite set of terminal variables,
- $R \in N \times (N \cup \Sigma)^*$ is a finite set of production rules, and
- S is the start symbol.

A PCFG $\mathcal{G} = (N, \Sigma, R, S, P)$ requires an extra element P which is a set of probabilities on each production rule. P is a function that maps each rule to a numerical probability such that the probabilities of rules with the same left-hand sides add up to 1.

2.1.2 Inside

Given a sentence $x_1, ...x_n$, the inside probability $\alpha(A, i, j)$ represents the total probability of generating words $x_{i:j}$ given the root nonterminal A. For all $1 \leq i \leq j \leq n$, the inside probability $\alpha(A, i, j)$ is computed as:

- Initialize: if $A \rightarrow x_i \in R$, then $\alpha(A, i, i) = p(A \rightarrow x_i)$, otherwise 0.
- From bottom to top:

$$\alpha(A, i, j) = \sum_{A \rightarrow BC} \sum_{k=i}^{j-1} p(A \rightarrow BC) \cdot \alpha(B, i, k) \cdot \alpha(C, k+1, j)$$

2.1.3 Outside

Given a sentence $x_1, ...x_n$ and inside probability, the outside probability $\beta(A, i, j)$ is the total probability of generating the context $x_{1:i-1} A x_{j+1:n}$ around an A over $x_{i:j}$. For all $1 \leq i \leq j \leq n$, the outside probability $\beta(A, i, j)$, is computed as:

- Initialize: if $A = S$, then $\beta(A, 1, n) = 1$, otherwise 0.
- From top to bottom:

$$\beta(A, i, j) = \sum_{B \rightarrow AC} \sum_{k=j+1}^{n} p(B \rightarrow AC) \cdot \beta(B, i, k) \cdot \alpha(C, j+1, k) + \sum_{B \rightarrow CA} \sum_{k=1}^{i-1} p(B \rightarrow CA) \cdot \beta(B, k, j) \cdot \alpha(C, k, i-1)$$

2.1.4 EM algorithm

For a corpus, $\mathcal{C}$, that consist of sentence $x_1, ...x_n$, we can then define the following:

Total probability of all trees:

$$Z = \sum_{x \in \mathcal{C}} \alpha(S, 1, \text{len}(x))$$

Total probability of contain $(A \rightarrow BC, i, j, k)$:

$$\mu(A \rightarrow BC, i, j, k) = \sum_{x \in \mathcal{C}} \sum_{i \leq k < j} p(A \rightarrow BC) \beta(A, i, j) \alpha(B, i, k) \alpha(C, k+1, j)$$

Total probability of containing A from $x_{i:j}$:

$$\mu(A, i, j) = \sum_{x \in \mathcal{C}} \alpha(A, i, j) \beta(A, i, j)$$

Total probability of A is the preterminal of x_i:

$$\mu(A, i) = \mu(A, i, i)$$

The detailed EM estimation of the parameters of a PCFG is demonstrated in Algorithm 1.

Algorithm 1 EM for PCFG

```
Require: a time step limit $T$, a list of sentences in the new corpus, and a PCFG $\mathcal{G} = (N, \Sigma, R, S, P)$.
1:  for $t \in$ range($T$) do
2:      $f(r) = 0$ for each rule $r$
3:      for each sent in sentences do
4:          compute $\alpha(), \beta(), \mu()$
5:          $f(A \rightarrow x) += \Sigma_{x_i = x} \frac{\mu(A,i)}{Z}$
6:          $f(A \rightarrow BC) += \Sigma_{i,j,k} \frac{\mu(A \rightarrow BC,i,j,k)}{Z}$
7:      end for
8:      for each rule $A \rightarrow \gamma$ do
9:          $P(A \rightarrow \gamma) = \frac{f(A \rightarrow \gamma)}{\Sigma_{A \rightarrow \gamma'} f(A \rightarrow \gamma')}$
10:     end for
11: end for
12: return $P$
```

2.2 Probabilistic Lambek Categorial Grammar

2.2.1 Proof Net

The product-free LCG (Lambek, 1958) builds categories over a set of primitives $\{p_1, p_2, ...\}$ with two binary connectives / and \. One can regard a category like S/NP as a function that takes an NP on the right to yield an S.

Given an LCG sequent, each derivation has a bijective, corresponding proof net. Proof nets can be used to determine whether a sequent is derivable. Roorda (1991) demonstrated that a proof net can be constructed from a sequent with the following steps.

Labeling Category Label each LHS category with a negative-polarity variable and label the RHS category as a positive-polarity variable. Each variable can only be used to label one LHS or RHS category.

Unfolding Categories Apply four substitution rules to categories recursively until all categories are unfolded into strings of signed primitives.

$$(A\backslash B)^{-}{:}t \rightarrow A^{+}{:}u \;\; B^{-}{:}tu \tag{1}$$
$$(A\backslash B)^{+}{:}v \rightarrow B^{+}{:}v' \;\; A^{-}{:}u[v := \lambda u.v'] \tag{2}$$
$$(A/B)^{-}{:}t \rightarrow A^{-}{:}tu \;\; B^{+}{:}u \tag{3}$$
$$(A/B)^{+}{:}v \rightarrow B^{-}{:}u \;\; A^{+}{:}v'[v := \lambda u.v'] \tag{4}$$

In the case of either rule (2) or (4), we will refer to v as a *lambda node*. It is the label on a positively signed complex category.

Add Linkages Link pairs of identical axiom primitives with opposite polarities. These linkages cannot be crossed with each other (i.e. they form a half-planar graph).

Variable Substitution Substitute variables according to the linked pairs until no further variables can be substituted.

2.3 LC-Graph

LC-graph (Penn, 2004) is a directed graph $\langle V, E\rangle$ where V is a finite set of variables that are used to label categories. E contains two kinds of edges:

- For all v, u, v' in rules (2) and (4), add (v, u) and (v, v') into E.
- For every axiom link that matches $p^{+} : u$ and $p^{-} : t$, where t is a string of variables, for all $v \in t$, add (u, v) into E.

LC-graphs provide a graph-theoretical way of determining the derivability of a sequent. A sequent is derivable if there exists an LC-graph such that:

- **I(0)** there exists a unique node that is path-accessible to all other nodes.
- **I(1)** G is acyclic.
- **I(2)** For all v, u, v' in rules (2) and (4), v' is path accessible to u.
- **I(CT)** For every lambda node v, there is a non-lambda node y and a terminal node x labelling a LHS category such that $v \rightsquigarrow y \rightarrow x$.

2.3.1 LC-Tree

Zhao and Penn (2024) showed that, for generation proposes, every proof net can be represented as a tree structure called an LC-tree. There is a mechanical process that can transform any proof net with its LC-graph into an LC-tree. Labels of nodes and edges called *augments* are added during this process.

For each edge (v, v'):

1. If v is a positive non-lambda node and v' is v's positive daughter, add as the edge's augment the connective that was unfolded by the substitution rule that created v'.

2. Otherwise, add nothing.

For each node u, define its augment, $\psi(u)$ such that:

1. If u is a lambda node, $\psi(u) = \lambda$.

2. If u is a positive non-lambda node, then there must be exactly one axiomatic formula $p^{+} : u$. Let $\psi(u) = p$.

3. If u is a negative non-lambda node, then there must be exactly one axiomatic formula $p^{-} : t$, where t is a string and $u \in t$. Let $\psi(u) = p$

They also proved that this transformation is bijective.

Consider the sequent of *"We are surprised"*. with sequent:

$$NP \quad (NP\backslash S)/(NP\backslash S) \quad NP\backslash S \models S$$

Figure 1 is its corresponding proof net. Figure 2 is its LC-graph. Figure 3 is its LC-tree, where node augments are shown in place of the nodes, and edge augments are shown as labels on their edges. This tree structure can be used to generate a sequent from an empty start using PCFG-style production rules.

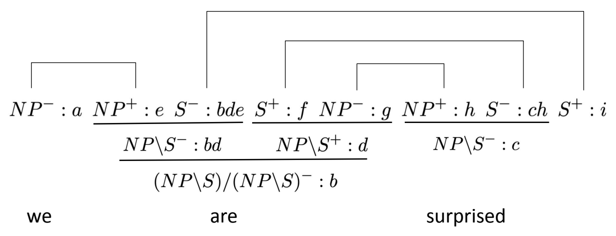

$NP^-:a$ $NP^+:e$ $S^-:bde$ $S^+:f$ $NP^-:g$ $NP^+:h$ $S^-:ch$ $S^+:i$
$NP\backslash S^-:bd$
$NP\backslash S^+:d$
$NP\backslash S^-:c$
$(NP\backslash S)/(NP\backslash S)^-:b$
we
are
surprised

Figure 1: Proof net example.

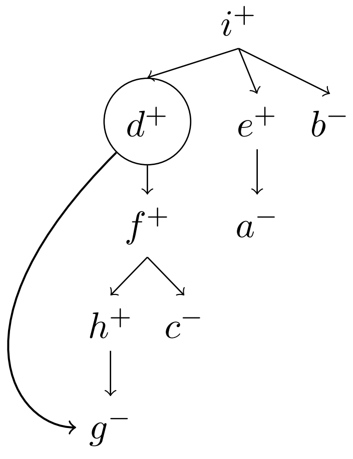


Figure 2: LC-graph example.

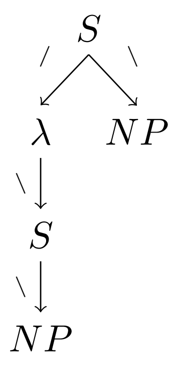


Figure 3: LC-tree example.

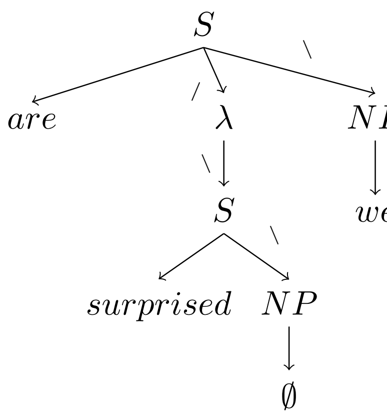


Figure 4: Generating lexical items.

3 Method

3.1 Adding Lexical Entries

Zhao and Penn (2024) demonstrated the LC-tree correspondence only for LCG-sequent generation, which means that the input LC-tree does not have words. In Figure 3, the (negative) variables labelling the LHS categories of the proof net are missing. Each of these can be assigned a lexical item during generation. For each negative variable that is a daughter of a lambda node (for example, node g in Figure 2), we assign it the special empty symbol, "λ." Figure 4 shows the tree after the words have been added.

3.2 Proof of Contiguity

Both inside and outside rely upon the label of a subtree to represent a contiguous span $x_i, ..., x_j$. We will now prove that such an assumption also holds in an LC-tree. Thus we can compute probabilities by splitting strings into substrings.

Lemma 3.1. *The transformation from proof net to LC-tree is bijective.*

Proof. Theorem 3.1 in Zhao and Penn (2024). □

Theorem 3.2. *The lexical items that are terminals of any subtree in an LC-tree must be contiguous in the original sequent.*

Proof. We first introduce a way to construct an LCG sequent from an LC-tree from bottom to top.
Base: For an LC-tree containing a single node p, construct the LCG sequent $p \models p$.
Otherwise:
case 1: If the root of the LC-tree is λ, there is a sequent $A_1, ...Ak \models A_{k+1}$ that can be constructed from the subtree that is rooted at λ's only child. A new sequent $A_2, ..., A_k \models A_1\backslash A_{k+1}$ or $A_1, ...A_{k-1} \models A_{k+1}/A_k$ can be constructed based on the edge augment.
case 2: If the root of the LC-tree is p where p is a primitive, then for each subtree $\mathcal{T}_i$, there exists a corresponding LCG sequent. We start from an initial sequent $p \models p$ and then iteratively construct new sequents from the left-most child to the right-most child. Assume that $\mathcal{T}_i$ can generate sequent $B_1, ...B_n \models B_{n+1}$. $A_1, ...A_m \models A_{m+1}$ is the already generated sequent from $\mathcal{T}_1$ to $\mathcal{T}_{i-1}$. The next step is to combine two sequents into one sequent. Assume A_k is the category

that contains the initial primitive p. Then either $A_1, ..., A_k/B_{n+1}, B_1, ...B_n, ...A_m \models A_{m+1}$ or $A_1, ..., B_1, ...B_n, B_{n+1}\backslash A_k, ...A_m \models A_{m+1}$ can be constructed based on the augment that is stored in the edge from p to $\mathcal{T}_i$.

By Lemma 3.1, this final sequent is the only sequent that corresponds to this LC-tree. Neither case will insert a new category in the middle of the subsequent. The new sequent is obviously contiguous since the entire LHS must be contiguous. □

3.3 Unify Internal Lambda Nodes

Both inside and outside probability assume that in a parse tree $\mathcal{T}$, certain $x_i, ...x_j$ can be derived from some unique symbol A. Unary internal rules would break this uniqueness. For example, in Figure 4, both λ and S can be the internal representation of span $(3, 3)$. To address this, we reconstruct the tree by unifying every unary node with its only child. This step will increase the number of non-terminal symbols. Algorithm 2 shows how to unify lambda nodes recursively from bottom to top. Figure 5 shows the result of unifying λ and its only child S from Figure 4 into a single node $\lambda\backslash S$.

Algorithm 2 Unify Nodes

```
Require: A LC-tree $\mathcal{T}$.
1: for each child in $\mathcal{T}$.children do
2:     child = unify_node(child)
3: end for
4: if $\mathcal{T}$.symbol = $\lambda$ then
5:     c = $\mathcal{T}$.child[0]
6:     c = $\mathcal{T}$.symbol + $(\mathcal{T}, c)$.connective +
7:         c.symbol
8: end if
9: return $\mathcal{T}$
```

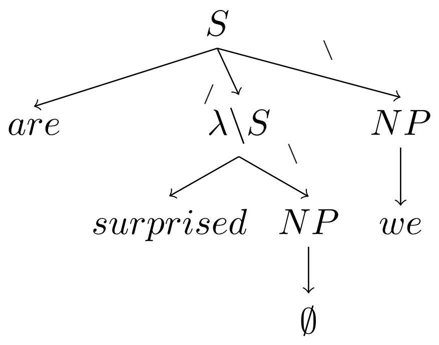


Figure 5: Unify internal lambda nodes.

3.4 Inside-Outside Algorithm

3.4.1 Initial Model

The inside-outside algorithm requires an initial model to improve. For the sake of concreteness, we will assume that this model follows the form outlined by Zhao and Penn (2024), in which there are two kinds of rules: $A \rightarrow x$, where A is a non-terminal and x is a terminal, and $A \rightarrow B_1 \dots B_m$, where A is a primitive, and the B_i are non-terminals.

In unified LC-trees, every local tree is of the form $A \rightarrow xB_1 \dots B_m$, where A is guaranteed to contain a unique primitive category. We obtain the rules of the PLCG by splitting each of these local trees into $\pi(A) \rightarrow x$ and $\pi(A) \rightarrow B_1 \dots B_m$ (when $m = 0$, we denote this rule as $\pi(A) \rightarrow \epsilon$), where $\pi(A)$ is the unique primitive contained in A. For instance, $\pi(/\lambda\backslash S) = S$.

In Figure 5, for example, the internal node $/\lambda\backslash S$ splits into $S \rightarrow surprised$ and $S \rightarrow \backslash NP$.

Zhao and Penn (2024) prescribes the following maximum likelihood estimates for these rules:

$$p(A \rightarrow B_1...B_m) = \frac{count(A \rightarrow B_1...B_m)}{count(A)}$$
$$p(A \rightarrow w) = \frac{count(A \rightarrow w)}{count(A \text{ in lexicon})}$$

As a toy example, for the corpus with only the one sequent in Figure 1, which has the one LC-tree in Figure 5, this initial model would be:

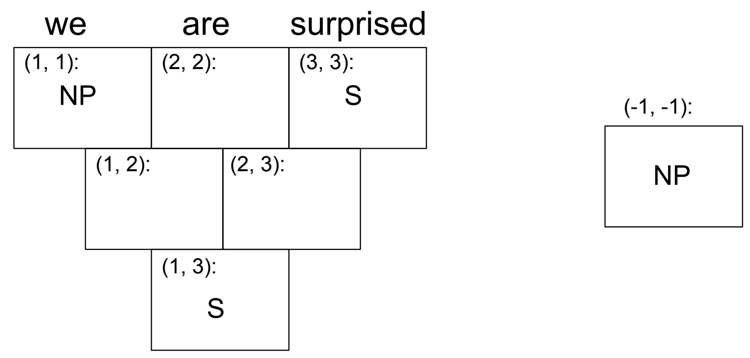
$$\begin{aligned} &Primitive\ rules: \\ &\qquad S \rightarrow /\lambda\backslash S,\ \backslash NP\ \ 0.5 \\ &\qquad S \rightarrow \backslash NP\ \ 0.5 \\ &\qquad NP \rightarrow \epsilon\ \ 1.0 \\ &Lexicon\ rules: \\ &\qquad NP \rightarrow we\ \ 1.0 \\ &\qquad S \rightarrow are\ \ 0.5 | surprised\ \ 0.5 \end{aligned}$$

3.4.2 Inside

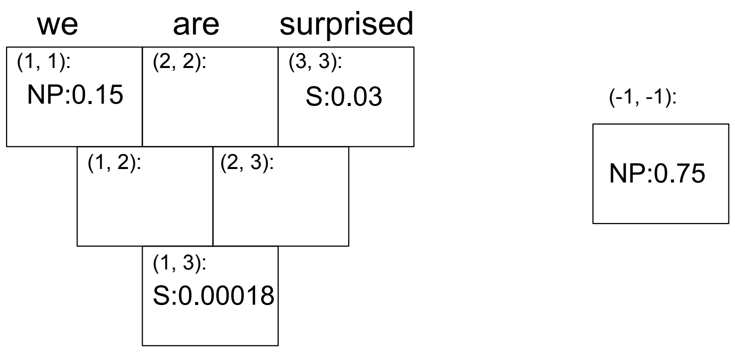
The inside probability $\alpha(A, i, j)$ represents the probability that A is the root of a subtree from word i to word j. Inside probabilities are once again computed from bottom to top:

Initialize: $\alpha(A, i, i) = p(A \rightarrow \epsilon)$
Otherwise:

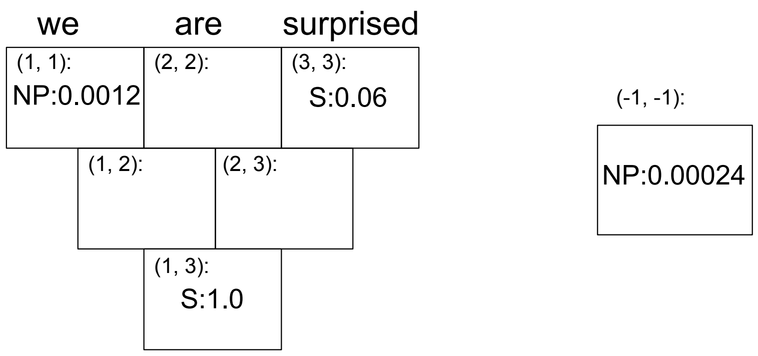
$$\alpha(A, i, j) = \sum_{\substack{A \rightarrow wB_1..Bm \\ s_1,...,s_m \\ e_1,...,e_m}} p(A \rightarrow w) \cdot p(A \rightarrow B_1..Bm) \prod_{k=1}^{k=m} \alpha(B_s, s_k, e_k),$$



(a) Chart parser that stores the symbol representation of $x_i, ..., x_j$



(b) Inside probability $\alpha(A, i, j)$



(c) Outside probability $\beta(A, i, j)$

$$\begin{aligned} f(NP \rightarrow \epsilon) &= 6 & p(NP \rightarrow \epsilon) &= 1.0 \\ f(S \rightarrow \backslash NP) &= 5 & p(S \rightarrow \backslash NP) &= 0.5 \\ f(S \rightarrow /L\backslash S, \backslash NP) &= 5 & p(S \rightarrow /L\backslash S, \backslash NP) &= 0.5 \end{aligned}$$

(d) EM estimation for PLCG.

Figure 6: Numerical example for inside-outside PLCG.

where s_k and e_k refer to the start(min) and end(max) lexical positions of the words rooted at B_k.

Some special terminal nodes in the LC-tree do not yield any lexical items. These are the negative daughters of lambda nodes. For example, one of the NPs in Figure 5 yields $\emptyset$. We will use the special additional base case $\alpha(A, -1, -1) = p(A \rightarrow \epsilon)$

	len≤ 10	len≤ 20	len≤ 30
before	7.2e-13	1.8e-13	1.1e-13
after	1.8e-12	2.8e-13	1.8e-13

Table 1: Average **sentence** probability in the test set before and after inside-outside PLCG.

	len≤ 10	len≤ 20	len≤ 30
before	0.000547	0.00014	8.4e-05
after	0.001522	0.00034	2.0e-04

Table 2: Average LCG **sequent** probability in the test set before and after inside-outside PLCG.

and terminal case $\beta(A, -1, -1)$ to store inside and outside probabilities for these categories.

3.4.3 Outside

Given a sentence $x_1, ...x_n$ and inside probabilities, the outside probability $\beta(A, i, j)$ is the total probability of generating words $x_{1:i-1}Ax_{j+1:n}$. Inside probabilities are computed from top to bottom:

Initialize: $\beta(S, 1, n) = 1$
Otherwise:

$$\beta(A,i,j) = \sum_{\substack{B\to wC_1..A..C_m \\ s_1,...,s_m \\ e_1,...,e_m}} \Big\{ p(B \to w) \cdot p(B \to C_1..A..C_m) \cdot \beta(B, s_B, e_B) \prod_{k=1}^{k=m} \alpha(C_s, s_k, e_k) \Big\},$$

where s_k and e_k refer to the start(min) and end(max) lexical positions of the words rooted at C_k. Also, s_B and e_B are the start and end of B.

3.4.4 EM algorithm

Similarly to PCFG, the below values have special meanings:

Total probability of all trees:

$$Z = \sum_{x\in\mathcal{C}} \alpha(S, 1, \text{len}(x))$$

Total probability of trees that contain $(A \to B_1...B_m, s_1, ..., s_m, e_1, ..., e_m)$:

$$\mu(A \to B_1...B_m, s_1, ..., s_m, e_1, ..., e_m) = \sum_{\substack{x\in\mathcal{C} \\ s_1,...,s_m \\ e_1,...,e_m}} p(A \to B_1...B_m)\beta(A, s_A, e_B) \cdot \prod_{k=1}^{k=m} \alpha(B_s, s_k, e_k)$$

Algorithm 3 shows the EM algorithm for re-estimating a PLCG. In line 1, we do parsing at the beginning and store derivations in the form of a chart parser. The initialization of the outside probability $\beta(S, 1, n) = 1$ ensures that only subtrees of the successful parses contribute to the parameter updates. So there is no need to store internal failed subtrees.

Algorithm 3 EM for PLCG

```
Require: The time-step limit $T$, and the training data that consist of the list of sentences with supertags.
1: Parse all the sentences with the given categories.
2: for $t \in$ range($T$) do
3:     $f(r) = 0$ for each rule $r$
4:     for each sent in sentences do
5:         compute $\alpha(), \beta(), \mu()$
6:         $f(A \to B_1...B_m) += $
7:         $\sum \frac{(A\to B_1...B_m, s_1,...,s_m, e_1,...,e_m)}{Z}$
8:     end for
9:     for each rule $A \to \gamma$ do
10:        $P(A \to \gamma) = \frac{f(A\to\gamma)}{\Sigma_{A\to\gamma'} f(A\to\gamma')}$
11:    end for
12: end for
13: return $P$
```

3.5 Example

Assume we start from a PLCG with the following rules:

$$
\begin{aligned}
S &\rightarrow /S \quad 0.2\\
S &\rightarrow \backslash\lambda/\lambda\backslash S, /NP, \backslash NP \quad 0.2\\
S &\rightarrow /NP, \backslash NP \quad 0.2\\
S &\rightarrow /L\backslash S, \backslash NP \quad 0.2\\
S &\rightarrow \backslash NP \quad 0.2\\
NP &\rightarrow /NP \quad 0.25\\
NP &\rightarrow \epsilon \quad 0.75\\
NP &\rightarrow we \quad 0.2\\
S &\rightarrow are \quad 0.2\\
S &\rightarrow surprised \quad 0.2\\
&\ldots
\end{aligned}
$$

and want to update the parameters on the corpus that contains one sentence: *"we are surprised."*

Figure 6a shows the CKY-style table which contains the symbol that forms the span $(x_i, ..., x_j)$. i and j are shown as tuples in these figures. Figure 6b and Figure 6c demonstrate how the inside and outside probabilities are computed. The final updated probabilities of the production rules are listed in Figure 6d. Recall that in PLCG, the negative daughters of lambda nodes do not generate any span. We use $(-1, -1)$ to represent these.

4 Results

4.1 Dataset

The Penn TreeBank (Marcus et al., 1993) is a human annotated tree bank. CCGbank (Hockenmaier and Steedman, 2007) is a conversion of the Penn TreeBank to combinatory categorial grammar. LCGbank (Bhargava et al., 2024), however, is a conversion and partial reannotation of CCGbank to LCG. We use the LCGbank training set (Sections 1-22 and 24) to initialize a PLCG and then run the inside-outside algorithm on the test set (Section 23). Note that we also include gold-standard lexical categories as a part of the input.

4.2 Sentence Generation

The average probabilities of sentences of the test set are listed in Table 1. The probability of each sentence is computed by summing over the probabilities of its parse trees.

The result shows that the average probability of the test sentences has significantly ($p < 10^{-192}, x = 0.0293$) increased after the application of our algorithm. This notable improvement demonstrates the effectiveness of our approach, validating its ability to enhance sentence probability estimates.

4.3 Sequent Generation

One special case is using this to generate LCG sequents by eliminating all lexical entries. In other words, we can assume a universal lexicon in which $p(A \rightarrow w)$ or $p(B \rightarrow w)$ is alsways 1 in computing both the inside and outside probabilities.

Table 2 shows the average sequent probability before and after the inside-outside algorithm. We can see that the average probability increases after applying the inside-outside algorithm across all lengths of sequents, with a significant difference ($p < 10^{-70}, x = 0.194$). This further confirms the robustness of our method in improving probability estimation even in specialized scenarios like LCG sequent generation.

5 Conclusion and Future Work

In this paper, we presented a novel inside-outside algorithm for PLCG. Our evaluation, however, still relies on gold-standard supertags from a human-annotated corpus. Removing this reliance and reconciling this inside-outside algorithm with Fowler (2007)'s polynomial time LCG-parsing algorithm, which uses a special, residual parsing chart representation, remain as future work.

Limitation

Our limitation is that the complexity of our algorithm is very dependent on the complexity of parsing. The LCG derivability problem has been proven to be NP-complete in the worst case.

References

Kenza Amara, Rita Sevastjanova, and Mennatallah El-Assady. 2024. SyntaxShap: Syntax-aware explainability method for text generation. In *Findings of the Association for Computational Linguistics ACL 2024*, pages 4551–4566, Bangkok, Thailand and virtual meeting. Association for Computational Linguistics.

Aditya Bhargava, Timothy A. D. Fowler, and Gerald Penn. 2024. LCGbank: A corpus of syntactic analyses based on proof nets. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pages 10225–10236, Torino, Italia. ELRA and ICCL.

Timothy AD Fowler. 2007. A polynomial time algorithm for parsing with the bounded order lambek calculus. In *Conference on Mathematics of Language*, pages 36–43. Springer.

Julia Hockenmaier and Mark Steedman. 2007. CCGbank: A corpus of CCG derivations and dependency structures extracted from the Penn Treebank. *Computational Linguistics*, 33(3):355–396.

T. Huang and K.S. Fu. 1971. On stochastic context-free languages. *Information Sciences*, 3(3):201–224.

Hiroshi Kanayama, Yang Zhao, Ran Iwamoto, and Takuya Ohko. 2024. Incorporating syntax and lexical knowledge to multilingual sentiment classification on large language models. In *Findings of the Association for Computational Linguistics ACL 2024*, pages 4810–4817, Bangkok, Thailand and virtual meeting. Association for Computational Linguistics.

Joachim Lambek. 1958. The mathematics of sentence structure. *The American Mathematical Monthly*, 65(3):154–170.

Karim Lari and Steve J Young. 1990. The estimation of stochastic context-free grammars using the inside-outside algorithm. *Computer speech & language*, 4(1):35–56.

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.

Raphaël Millière. 2024. Language models as models of language. *Preprint*, arXiv:2408.07144.

Jingcheng Niu, Wenjie Lu, and Gerald Penn. 2022. Does BERT rediscover a classical NLP pipeline? In *Proceedings of the 29th International Conference on Computational Linguistics*, pages 3143–3153, Gyeongju, Republic of Korea. International Committee on Computational Linguistics.

Gerald Penn. 2004. A graph-theoretic approach to sequent derivability in the lambek calculus. *Electronic Notes in Theoretical Computer Science*, 53:274–295. Proceedings of the joint meeting of the 6th Conference on Formal Grammar and the 7th Conference on Mathematics of Language.

Dirk Roorda. 1991. *Resource Logics: Proof-Theoretical Investigations*. Ph.D. thesis.

Behzad Shayegh, Yuqiao Wen, and Lili Mou. 2024. Tree-averaging algorithms for ensemble-based unsupervised discontinuous constituency parsing. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15135–15156, Bangkok, Thailand. Association for Computational Linguistics.

Yikang Shen, Zhouhan Lin, Chin-Wei Huang, and Aaron Courville. 2018. Neural language modeling by jointly learning syntax and lexicon. *Preprint*, arXiv:1711.02013.

Sixuan Wu, Jian Li, Peng Zhang, and Yue Zhang. 2021. Natural language processing meets quantum physics: A survey and categorization. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 3172–3182, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Jinman Zhao and Gerald Penn. 2021. A generative process for Lambek categorial proof nets. In *Proceedings of the 17th Meeting on the Mathematics of Language*, pages 1–13, Umeå, Sweden. Association for Computational Linguistics.

Jinman Zhao and Gerald Penn. 2024. A generative model for Lambek categorial sequents. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pages 584–593, Torino, Italia. ELRA and ICCL.