Improved Combinatory Categorial Grammar Induction with Boundary Words and Bayesian Inference

Yun HUANG[1,2] *Min ZHANG*[2] *Chew Lim TAN*[1]

[1]Department of Computer Science
National University of Singapore
13 Computing Drive, Singapore
`{huangyun,tancl}@comp.nus.edu.sg`

[2]Human Language Department
Institute for Infocomm Research
1 Fusionopolis Way, Singapore
`mzhang@i2r.a-star.edu.sg`

ABSTRACT
Combinatory Categorial Grammar (CCG) is an expressive grammar formalism which is able to capture long-range dependencies. However, building large and wide-coverage treebanks for CCG is expensive and time-consuming. In this paper, we focus on the problem of unsupervised CCG induction from plain texts. Based on the baseline model in (Bisk and Hockenmaier, 2012), we propose following two improvements: (1) we utilize boundary part-of-speech (POS) tags to capture lexical information; (2) we perform nonparametric Bayesian inference based on the Pitman-Yor process to learn compact grammars. Experiments on English Penn treebank demonstrate the effectiveness of our boundary model and Bayesian learning.

TITLE AND ABSTRACT IN ANOTHER LANGUAGE (CHINESE)

基于边界词和贝叶斯模型改进的组合范畴语法推导

组合范畴语法(CCG)是一种具有丰富表达能力的语法形式，它可以捕获长距离的依赖关系。但是，构建大规模、覆盖面广的组合范畴语法语料库既昂贵又耗时。在本文中，我们研究如何从普通文本中无监督地推导出组合范畴语法。基于现有的工作(Bisk and Hockenmaier, 2012)，我们提出以下两个改进：(1) 我们使用边界词的词性标记把词汇化信息引入模型中；(2) 我们使用基于Pitman-Yor过程的非参数贝叶斯模型来学习简洁的文法。在英语宾州树库上的实验结果显示了我们提出的边界词模型和贝叶斯模型的有效性。

KEYWORDS: Grammar Induction, Combinatory Categorial Grammar, Boundary Words, Bayesian Model.

KEYWORDS IN CHINESE: 语法推导, 组合范畴语法, 边界词, 贝叶斯模型.

1 Introduction

Unsupervised grammar induction has attracted research interests for a long time. The induced grammars can be used to construct large treebanks (van Zaanen, 2000), study language acquisition (Jones et al., 2010), etc. In recent years, numerous approaches have been introduced to automatically induce hierarchical structures from plain strings. Some approaches focus on the constituency grammar induction: the constituent-context model (Klein and Manning, 2002), the data-oriented parsing (Bod, 2006), the common cover link model (Seginer, 2007), and the tree-substitution grammars (TSG) (Cohn et al., 2009). The other mainstream is the dependency grammar induction: the dependency model with valence (DMV) (Klein and Manning, 2004; Headden III et al., 2009; Cohen and Smith, 2009), TSG model for dependency (Blunsom and Cohn, 2010), etc.

Among these grammar formalisms, the Combinatorial Categorial Grammar (CCG) is a lexicalized, mildly-context-sensitive model (Steedman, 2000). In the formal grammar theory, CCGs are known to be weekly equivalent to Linear Indexed Grammars, Tree-adjoining Grammars, and Head Grammars (Vijay-Shanker and Weir, 1994). The CCG formalism provides a transparent interface between syntax and semantics, such that the underlying semantics could be naturally defined over syntactical derivations, including the long-range dependencies, the coordination structure, and the extraction phenomenon (Bos et al., 2004; Zettlemoyer and Collins, 2007). As a mildly context-sensitive grammar, CCG can be efficiently parsed in polynomial time, which makes them practical in real parsing tasks[1]. The wide-coverage combinatorial categorial grammars have been used in many NLP tasks, such as the lexical acquisition (Blunsom and Baldwin, 2006), the parsing tasks (Hockenmaier and Steedman, 2002; Clark and Curran, 2003), and the statistical machine translation (Hassan et al., 2007; Zhang and Clark, 2011). These supervised CCG models highly depend on the annotated training corpus, e.g. the CCGbank (Hockenmaier and Steedman, 2007). However, building large and wide-coverage treebanks for CCG is expensive and time-consuming. Therefore, how to induce CCG lexicons and grammars from unlabeled sentences has great values.

Some unsupervised CCG induction models have been proposed (Osborne and Briscoe, 1997; Watkinson and Manandhar, 1999; Ponvert, 2007; Boonkwan and Steedman, 2011; Bisk and Hockenmaier, 2012). Most of these approaches define probabilistic models over CCG rules and use the Expectation-Maximization (EM) algorithm to estimate parameters. The generative process generates grammar rules independently given their parents, without regard to the lexical information. However, the constituents and contexts have been proven useful for grammar induction (Klein and Manning, 2002; Headden III et al., 2009). Another issue of the EM-based models is that the EM algorithm tends to overfit the training data, which requires carefully smoothing (Headden III et al., 2009).

In this paper, we propose to incorporate the lexical information in the unsupervised CCG induction in order to capture more complex language aspects. Specifically, an additional *boundary model*, which defines probability distributions over the boundary part-of-speech tags, is introduced during the probability calculation for parse trees. Furthermore, we present the nonparametric Bayesian inference to alleviate the overfitting problem of EM. The Pitman-Yor process (Pitman and Yor, 1997) is used to encourage rule reuse, resulting in compact grammars. Although the boundary words and Bayesian inference have been used in other grammar

[1]Given grammar G, the parsing complexity of CCG is $O(n^3|G|)$ for the sentences with length n. Clark and Curran (2007) report their supervised parser could parse $20-30$ sentences/second using the treebank grammar.

induction models, so far as we know they are used in CCG induction for the first time. Experimental results show that both the boundary model and the Bayesian inference outperform the baseline CCG induction system significantly.

This paper is structured as follows. First we give a brief overview of the combinatorial categorial grammars in Section 2. Then we present the grammar generation step in Section 3. In Section 4, we describe the baseline model and propose the boundary model and the non-parametric Bayesian learning framework. Experimental results and related work are shown in Section 5 and Section 6 respectively. We conclude our work in the final section.

2 Combinatory Categorial Grammar

Combinatory Categorial Grammar (CCG) is a linguistically expressive lexicalized grammar formalism (Steedman, 2000). In CCG, words and nonterminals are associated with rich syntactic categories which capture the basic word order and subcategorization. Specifically, the categories in CCG are defined recursively: (1) There are some atomic categories, e.g. S, N; (2) Complex categories take the form X/Y or X\Y, representing the syntactical function that takes the input category Y and outputs the result category X. The forward slash (/) and the backward slash (\) indicate the input category Y follows or precedes the complex category respectively. Note that X and Y themselves may be complex categories too. Parentheses can be used to specify the order of function applications if needed. By default, the slashes are left-associated, e.g. "X\Y/Z" is the shorthand of "(X\Y)/Z". If the order of categories is not important in some cases, we use the symbol "|" to represent either the forward slash or the backward slash. The following examples show some common categories in English grammars: S for sentences, N for nouns and noun phrases[2], (S\N)/N for transitive verbs, N/N for determiners and adjectives, etc.

The *derivation* of CCG is the sequence of CCG rule applications. There are a few kinds of rule templates defined in CCG. The simplest rules are the forward application (>) and the backward application (<), where the complex category consumes the whole input category:

$$\begin{array}{ccccc} \mathtt{X/Y} & \mathtt{Y} & \Rightarrow & \mathtt{X} & (>) \\ \mathtt{Y} & \mathtt{X\backslash Y} & \Rightarrow & \mathtt{X} & (<) \end{array}$$

The input category could be complex category as well, forming the composition rules:

$$\begin{array}{ccccc} \mathtt{X/Y} & \mathtt{Y|Z} & \Rightarrow & \mathtt{X|Z} & (>B^1) \\ \mathtt{Y|Z} & \mathtt{X\backslash Y} & \Rightarrow & \mathtt{X|Z} & (<B^1) \end{array}$$

Note that the above composition rules could reduce the categories X/Y and Y\Z to the category X\Z, using the so-called *cross composition rule*. These rules give CCG the ability to deal with the crossed dependencies in some languages such as Dutch or German[3].

Higher order composition rules can be defined similarly:

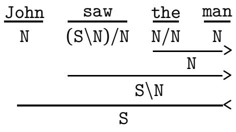
$$\begin{array}{ccccc} \mathtt{X/Y} & \mathtt{Y|Z_1|\ldots|Z_n} & \Rightarrow & \mathtt{X|Z_1|\ldots|Z_n} & (>B^n) \\ \mathtt{Y|Z_1|\ldots|Z_n} & \mathtt{X\backslash Y} & \Rightarrow & \mathtt{X|Z_1|\ldots|Z_n} & (<B^n) \end{array}$$

In a sense, the application rules (> and <) can be regarded as the zero-order case of composition rules (>B^0 and <B^0).

[2]In formal English grammars, NP is often used to represent noun phrases(Hockenmaier and Steedman, 2007). Following (Bisk and Hockenmaier, 2012), we do not distinguish noun phrase from nouns in this paper for efficiency. This simple treatment causes some problems, e.g. the determiners would be treated as adjuncts and then regarded optional, but actually they are needed for singular count nouns. We leave this problem to future work.

[3]See the example 71 (a German sentence) in (Steedman and Baldridge, 2011)

The following examples show the CCG derivations of a declarative sentence:



(1)

In this example, the lexical category (S\N)/N for the transitive verb "saw" restricts that the verb must first consume an object noun (N) on the right to become the intransitive verb category S\N, then take another noun (N) on the left as the subject to form the sentence S. We can see that the lexicons encode rich lexical information as well as the syntactic restrictions.

For coordination, only the same categories can be conjuncted to yield a single category of the same type in CCG. In detail, the CCG includes a ternary conjunction rule (&).

X	conj	X	⇒	X	(&)

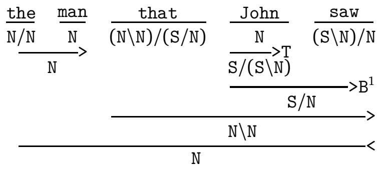
For the parsing algorithms (e.g. the bottom-up CKY algorithm) that require binary rules, we often use the binarized conjunction rules (>& and <&):

X	X[conj]	⇒	X	(>&)
conj	X	⇒	X[conj]	(<&)

The type-raising rules are also included in CCG, which turn arguments into functions over functions-over-such-arguments:

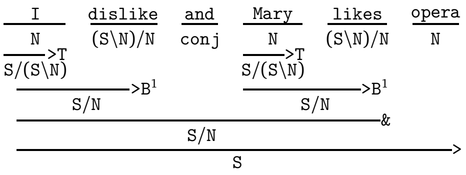
X	⇒	T/(T\X)	(>T)
X	⇒	T\(T/X)	(<T)

These rules are needed to form some unusual constituents, such as the constituent "John saw" in the example (2). In this example, there is no argument on the right to transitive verb "saw" due to the clause structure, so the noun "John" has to be type-raised.



(2)

Another example of type-raising is the uncommon coordination case (example (3)), in which the two uncommon subject-verb categories S/N are conjuncted.



(3)

In example (1) and (2), it should be emphasised that the same words have the same lexical categories, although the sentence structures are totally different. This elegant treatment for coordination and extraction structures in CCG allows easy recovery of the long-range dependencies and semantics.

Following (Bisk and Hockenmaier, 2012), we group the complex categories into the following two types according to their linguistic functions.

- **Modifier category**. Categories in the form of `X|X` are modifier categories. In other words, modifier categories take one category as input and output the same category as result. Some modifier category examples are the noun modifier "`N/N`", the sentence modifier "`S\S`", and the more complex category "`(S\S)/(S\S)`".
- **Functor category**. In contrast, the functor category takes one category as input but output a different category as result, i.e. in the form of `X|Y`, where `X` and `Y` are different categories. In the example (2), the follows are all functor categories: the transitive verb "`(S\N)/N`", the uncommon constituent "`S/N`", the type-raised category "`S/(S\N)`", and the relative pronoun "`(N\N)/(S/N)`".

The modifier categories and the functor categories play different linguistic roles. Using the dependency terminology, the modifier category `X|X` acts as the dependent and modifies its head `X`, while the functor category `X|Y` acts as the head of its dependent `Y`. We will revisit this issue in the CCG evaluation (section 5.1).

In this paper, we focus on the problem of unsupervised CCG induction, the task to infer meaningful grammars and tree structures from plain texts. We will first describe the grammar generation step in Section 3, and propose the boundary model and the Bayesian learning method in Section 4.

3 Grammar Generation

The combinatory categorial grammars explicitly encode the head-modifier and head-argument dependencies into rich syntactic categories. The first step of our CCG induction method is the lexicon generation step. Bisk and Hockenmaier (2012) propose a simple iterative lexicon generation algorithm from the golden part-of-speech tags. Due to the simplicity and effectiveness of this method, we adopt it to generate lexicons in our method. We rephrase their algorithm with minor modifications in this section.

Only two atomic categories, `N` (nouns or noun phrases) and `S` (sentences) are allowed in the grammar. Conjunction words (usually with part-of-speech tag `CC` in the Penn Treebank) are expanded from a special conjunction category `conj`. All trees are generated from a special start symbol `TOP`. Following (Bisk and Hockenmaier, 2012), we assume all strings are either nouns or sentences, i.e. they are generated from one of the following two unary rules:

$$\mathtt{TOP} \rightarrow \mathtt{N} \qquad\qquad \mathtt{TOP} \rightarrow \mathtt{S}$$

In addition, we restrict that: (1) strings containing at least one verb must be sentences, i.e. parsed with the $\mathtt{TOP} \rightarrow \mathtt{S}$ rule; and (2) strings without any verb must be parsed with the $\mathtt{TOP} \rightarrow \mathtt{N}$ rule[4]. Note that the above assumptions are not always true for real sentences. For instance, there exist some sentence fragments and exclamatory sentences in the English treebank. In this paper, we just follow previous work and have no special consideration on these cases for simplicity.

[4]Only the first restriction is used in (Bisk and Hockenmaier, 2012).

The initial CCG lexicon $\mathscr{L}^{(0)}$ is created manually. We associate the noun POS tags with the atomic category N, the verb POS tags with the atomic category S, and the conjunction POS tag with the special category conj. For the POS tag set of the English Penn treebank (Marcus et al., 1993), the initial CCG lexicons are shown as follows:

$$\begin{aligned} \texttt{N} &: \{\texttt{DT, NN, NNS, NNP, NNPS, PRP}\} \\ \texttt{S} &: \{\texttt{MD, VB, VBD, VBG, VBN, VBP, VBZ}\} \\ \texttt{conj} &: \{\texttt{CC}\} \end{aligned}$$

Note that the tag NNPS (representing plural proper nouns) and the tag VBP (representing verbs of non-3rd person singular and in present tense) are missed in (Bisk and Hockenmaier, 2012), but these two tags are found in the Penn treebank tag set.

The lexicon for atomic categories remains fixed after the initial lexicon $\mathscr{L}^{(0)}$ has been created. However, the POS tags may acquire more syntactic categories in the lexicon generation stage. In each induction step, we first assign each POS tag with the categories induced in the previous step, then create new candidates for adjacent POS tags of the training sentences.

- **Modifier category**. Assuming the i^{th} POS tag in some given sentence has been associated with category X, we create new modifier category X/X and X\X, if X satisfies one of the following conditions (items with [c]):
  [c] X is an atomic category;
  [c] X is a modifier itself.
  The newly created categories are inserted to the candidate set of the $(i-1)^{th}$ POS tag and the $(i+1)^{th}$ POS tag respectively.
- **Functor category**. For each pair of adjacent POS tags in the i^{th} and the $(i+1)^{th}$ position in each training sentence, and for each category X and Y associated with the two POS tags, we consider that X may take Y as argument to form the functor category X/Y, and Y may also take X as argument resulting in the functor Y\X. The new categories are valid if the head H (input category) and the argument A (result category) satisfy one of the following conditions (items with [c]) and violate none of the following restrictions (items with [r]):
  [c] H is a modifier or in the form of "(S|...)", and A is the atomic category "N" or "S";
  [c] H is "S" and A is "N", i.e. the categories "S/N" and "S\N" are allowed;
  [r] A is not a modifier, i.e. any non-modifier (atoms and functors) may be argument;
  [r] H is different from A, otherwise the result category is the modifier category rather than the functor category;
  [r] H is not "N", since the atomic category "N" is assumed to take no arguments.
  If the categories X/Y (or Y\X) passes the above tests, it is inserted to the candidate set of the i^{th} POS tag (and the corresponding $(i+1)^{th}$ POS tag).

After creating the lexicons for one sentence, we parse it with the created lexicons and remove categories that can not lead to a successful parse. The rest categories are inserted to the lexicon $\mathscr{L}^{(i)}$ for the i^{th} induction step. We perform this step twice to get the final lexicon $\mathscr{L}^{(2)}$.

The above induction procedure is almost the same as the algorithm described in (Bisk and Hockenmaier, 2012). One additional induction step they used is the "derived" induction step, in which adjacent constituents that can be derived from the existing lexicon are combined. However, their experiments do not show significant improvement of this lexicon generation method, so we omit this step in our experiments.

4 Improved CCG Induction Models

4.1 Basic Probabilistic Model

The *basic* model in this paper is the baseline model described in (Hockenmaier and Steedman, 2002), which is also used in (Bisk and Hockenmaier, 2012). There are four types of CCG rules:

1. the lexical (W) rules which generate terminal words;
2. the unary (U) rules which could be the root rules or the type-raising rules;
3. the left-headed (L) rules with the first child symbol as the head category, e.g. the forward composition rules;
4. the right-headed (R) rules with the second child symbol as the head category, e.g. the backward composition rules.

Binary trees are generated top-down recursively from the start symbol TOP. For each unexpanded nonterminal P, the basic model first generates the expansion type $\texttt{exp} \in \{\texttt{W}, \texttt{U}, \texttt{L}, \texttt{R}\}$ according to $P_e(\texttt{exp}|\texttt{P})$. Then for each expansion type, the model generates either terminal word w or head child H and possible non-head child N:

$$\begin{aligned}
&\text{Lexical:} && P_e(\texttt{exp}=\texttt{W}|\texttt{P})\,P_w(\texttt{w}|\texttt{P},\texttt{exp}=\texttt{W})\\
&\text{Unary:} && P_e(\texttt{exp}=\texttt{U}|\texttt{P})\,P_U(\texttt{H}|\texttt{P},\texttt{exp}=\texttt{U})\\
&\text{Left:} && P_e(\texttt{exp}=\texttt{L}|\texttt{P})\,P_L(\texttt{H}|\texttt{P},\texttt{exp}=\texttt{L})\,P_l(\texttt{N}|\texttt{P},\texttt{H},\texttt{exp}=\texttt{L})\\
&\text{Right:} && P_e(\texttt{exp}=\texttt{R}|\texttt{P})\,P_R(\texttt{H}|\texttt{P},\texttt{exp}=\texttt{R})\,P_r(\texttt{N}|\texttt{P},\texttt{H},\texttt{exp}=\texttt{R})
\end{aligned}$$

After the lexicon generation step, each POS tag acquires a lexicon of CCG categories. These lexicons and CCG rules are used to parse the training corpus. We use the Expectation Maximization (EM) algorithm to estimate model parameters for the basic model. The Inside-Outside algorithm (Lari and Young, 1990) is used to collect the expected counts in the E-step of the EM algorithm.

4.2 Boundary Models

Boundary POS tags have been proven useful for detecting phrase boundaries in the supervised setting (Xiong et al., 2010) and in the unsupervised grammar induction (Golland et al., 2012; Huang et al., 2012). We introduce this idea to the unsupervised combinatory categorial grammar induction. Since the system inputs are the golden POS tags in the treebank, we use the boundary words and the boundary POS tags interchangeably in this paper.

Particularly, in some parse tree T, we consider the boundary POS tags of each constituent and define the new probabilistic model as

$$\begin{aligned}
P(T) &= P_{CCG}(T)P_{BDR}(T)\\
&= \prod_{\text{rule}:r\in T} P_{CCG}(r) \prod_{\text{span}:\langle i,j\rangle\in T} P_{BDR}(\sigma_{\langle i,j\rangle}|B)
\end{aligned} \tag{4}$$

where distribution P_{CCG} is the basic CCG model defined in section 4.1, P_{BDR} is the proposed boundary model, $\sigma_{\langle i,j\rangle}$ means the boundary words of the constituent covered by span $\langle i,j\rangle$, and B is a special nonterminal representing the constituent spans. We denote this model as *basic+bdr*. Figure 1 shows a tree example. The boundary probability of this parse tree is

$$\begin{aligned}
P_{BDR}(T) = {} & P(\texttt{DT\_DT}|B) \times P(\texttt{NNS\_NNS}|B) \times P(\texttt{VBD\_VBD}|B) \times P(\texttt{RB\_RB}|B)\\
& \times P(\texttt{DT\_NNS}|B) \times P(\texttt{VBD\_RB}|B) \times P(\texttt{DT\_RB}|B)
\end{aligned} \tag{5}$$

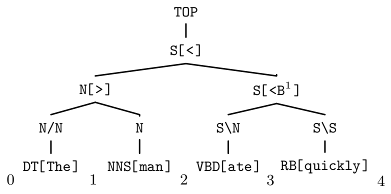


Figure 1: A tree example used to illustrate the boundary probability. The CCG rule types are given in the square brackets next to each nonterminal. Although only POS tags are considered in the induction model, we also show the words for clarity.

Currently, we restrict that the single special nonterminal B generates all boundary tag pairs. We have tried to let the boundary pairs depend on the category of tree nodes. For instance, for span $\langle 2, 4\rangle$ of tree in Figure 1, we model $P(\texttt{VBD\_RB}|S)$ rather than $P(\texttt{VBD\_RB}|B)$. However, this category-dependent boundary model performs poor in experiments (not reported in this paper). The reason might be the data sparsity problem, since there are quite a lot of induced categories in the grammar.

4.3 Bayesian Models

The EM algorithm may overfit the training data, so we propose the Bayesian model to infer grammars and tree structures. In the Bayesian models, the generative process is often formulated as the Chinese Restaurant process (CRP) or the Pitman-Yor process (PYP) to encourage rule reuse and learn compact models (Teh et al., 2006; Pitman and Yor, 1997). Since the PYP is a generalization of the CRP and has more elegant and controllable behavior over the "long tail" of probability distributions, we focus on the PYP in this paper.

For each nonterminal category A in CCG, we maintain a cache to store the total number n of rules expanded with A as parent, the total number of different rule types m, and the counts n_k of each rule that has been generated for $k = 1, \dots, m$. Initially, all caches are empty, i.e. with $n = m = 0$. The parse trees are generated in sequence. For each sentence, the PYP generates trees in the top-down fashion. For each nonterminal label to be expanded, we consult the cache associated with that nonterminal and decide whether to choose the k^{th} expanded rule in the cache, or generate a new rule. The probabilities of these two cases are

$$P_t(z|z_{i<n}) = \begin{cases} \frac{ma+b}{n+b} & \text{if } z_{n+1} = m+1 \\ \frac{n_k-a}{n+b} & \text{if } z_{n+1} = k, k \in \{1, \cdots, m\} \end{cases} \tag{6}$$

where z_i is the cache index of the i^{th} generated rule, $a \in [0, 1]$ and $b \geq 0$ are two category-associated parameters naming the discount and concentration parameters respectively. Note that different labels may have different values of a and b. If we decide to generate a new rule, the new rule is sampled from the base multinomial distribution P_0. We also put a Dirichlet prior on the base distribution and sample the base rule probabilities from the Dirichlet distribution: $\boldsymbol{\theta} \sim \text{Dir}(\boldsymbol{\theta}|\boldsymbol{\alpha})$. The above sampling procedures are performed recursively down until all frontier labels have been expanded to terminals. For CCG induction models described in previous sections, PYP priors are put on all factored models, although they may have different hyperparameters.

The joint probability of a particular sequence of indexes $\boldsymbol{z}$ with cached counts $(n_1, \ldots, n_m)$ under the Pitman-Yor process is

$$PY(\boldsymbol{z}|a, b) = \frac{\prod_{k=1}^{m}(a(k-1)+b)\prod_{j=1}^{n_k-1}(j-a)}{\prod_{i=0}^{n-1}(i+b)}. \tag{7}$$

The above generative process demonstrates the "rich get richer" dynamics, i.e. previous sampled rules would be sampled more likely in following procedures. It is easy to verify that any permutation of $z_1, \ldots, z_n$ has the same probability in the Pitman-Yor process, so the Pitman-Yor process is *exchangeable*, resulting in efficient sampling methods. Given the parse tree set $\mathscr{T}$, we could integrate out the base distribution probabilities to get the joint PYP probability[5]:

$$P(\mathscr{T}|\boldsymbol{\alpha}, \boldsymbol{a}, \boldsymbol{b}) = \prod_{\mathtt{X}\in\mathscr{N}} \frac{\text{Beta}(\boldsymbol{\alpha}_{\mathtt{X}} + \boldsymbol{f}_{\mathtt{X}})}{\text{Beta}(\boldsymbol{\alpha}_{\mathtt{X}})} PY(\boldsymbol{z}(\mathscr{T})|\boldsymbol{a}, \boldsymbol{b}) \tag{8}$$

where $\mathscr{N}$ is the set of nonterminal categories, $\boldsymbol{f}_{\mathtt{X}}$ is the vector containing the number of occurrences that rules r with $\mathtt{X}$ as parent in $\mathscr{T}$, and Beta means the Beta function.

To infer trees and parameters of the PYP model, we apply the collapsed Metropolis-Hastings algorithm (Hastings, 1970; Johnson et al., 2007) to sample trees from the parse forests. In detail, we iteratively draw samples for each yield in training corpus in sequence. Assuming the current tree of the i^{th} sentence is T_i, we first remove this tree from the whole tree set to obtain $\mathscr{T}_{-i}$, the set of sampled trees except the i^{th} one. Then we draw new tree T_i' from some proposal distribution $Q(T_i'|\mathscr{T}_{-i})$, and accept the new sampled tree with probability

$$A(T_i, T_i') = \min\left\{1, \frac{P(T'|\alpha, \boldsymbol{a}, \boldsymbol{b})\; Q(T_i|\mathscr{T}_{-i})}{P(T|\alpha, \boldsymbol{a}, \boldsymbol{b})\; Q(T_i'|\mathscr{T}_{-i})}\right\}. \tag{9}$$

In theory, Q could be any distribution if it never assigns zero probability. In practice, the proposal distribution should be close enough to the true distribution to avoid high rejection rate. We use the following proposal distribution in experiments:

$$Q(T_i|\mathscr{T}_{-i}) = \frac{1}{Z(\mathscr{T}_{-i})} \prod_{\text{rule}:r\in T_i} P_t(z_r|z_{\mathscr{T}_{-i}})\; P_0(r|\alpha)^{\delta(r\notin\mathscr{T}_{-i})} \tag{10}$$

in which P_t is the conditional index probability in Equation 6, and the model needs to consult the base distribution P_0 if it encounters a new rule ($\delta(r \notin \mathscr{T}_{-i}) = 1$). We do not need to calculate the normalization constant $Z(\mathscr{T}_{-i})$ since it would be cancelled in Equation 9. The proposal distribution differs from the true distribution in the sense that: the caches are updated immediately after calculating probabilities of each rule in T_i under the true distribution, while the caches stay fixed in the proposal distribution evaluation. In experiments, we observe that only a tiny fraction (less than 1%) of proposals are rejected. This provides evidence that the proposal distribution works well enough. We use the sampling algorithm described in (Blunsom and Osborne, 2008) to draw a parse tree from the parse forest according to the proposal distribution Q.

[5]For simplicity, we omit probability factorization as if there is only one model. The complete probability expression is the product of multiple factored PYP probabilities.

5 Experiments

5.1 Datasets and Settings

We carry out experiments on the Wall Street Journal portion of the Penn English Treebank (Marcus et al., 1993). As the standard data split, we use sections 02-21 as the training set, section 00 as the development set, and section 23 as the final test set. We remove punctuations and null elements in treebank, as the standard preprocessing step in the previous unsupervised grammar induction approaches (Klein and Manning, 2002; Cohn et al., 2010; Bisk and Hockenmaier, 2012). For comparison, we build datasets with sentence lengths no more than 10 and 20 words after removing punctuations. As the standard machine learning pipeline, we perform learning and inference on the training set, select model with best performance on the development set, and report the result of the selected model on the test set. For efficiency, we only train and tune parameters on sentences with length no more than 10, but report performance on longer sentences as well. Table 1 gives the statistics of each dataset.

Dataset	Train		Dev		Test	
	# sent	# word	# sent	# word	# sent	# word
PTB10	5,899	41,701	265	1,875	398	2,649
PTB20	-	-	-	-	1,286	16,591

Table 1: Data statistics

For evaluation, the script of CoNLL 2008 shared task[6] is used to calculate the Unlabeled Attachment Score (UAS) of the system outputs, using the treebank dependency structures as golden standards. We perform the McNemar's significant test to compare our proposed models with the baseline model. Since the original treebank only has the phrase-structure trees, we use (Johansson and Nugues, 2007)'s code[7] to convert the treebank to dependency structures. In order to compare with existing approaches, we follow (Bisk and Hockenmaier, 2012) to convert the CCG trees to dependency trees: (1) the modifier categories are treated as the dependents of their heads; (2) the head of the sentence is treated as a dependent of a root node at position 0; (3) the left part of conjunction is treated as the head of `conj`, and the `conj` is treated as the head of the right part. Figure 2 shows an example.

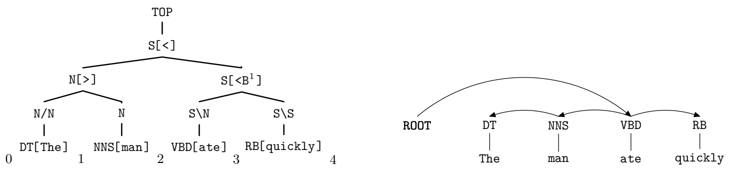


Figure 2: Left: a CCG tree example. Right: the converted dependency tree.

To reduce the model complexity, we restrict that the maximal order of composition rule is 2. The rule probabilities are initialized uniformly. For EM-based models (basic and basic+bdr), we add fixed value to expected counts in each E-step as smoothing. We perform maximal 40 EM iterations while stop earlier if the development score starts to drop. In the Bayesian

[6]Available at: `http://barcelona.research.yahoo.net/dokuwiki/doku.php?id=conll2008:software`
[7]Available at: `http://nlp.cs.lth.se/software/treebank_converter`

inference, we run sampler through the whole training sentences for 400 iterations and use the last sampled grammars to parse fresh sentences. To model the uncertainty of hyperparameters, we put an uninformative Beta$(1,1)$ prior on $\boldsymbol{a}$ and a "vague" Gamma$(10,0.1)$ prior on $\boldsymbol{b}$ instead of setting them empirically. After each iteration, we resample each of hyperparameters from the posterior distribution of hyperparameters using a slice sampler (Neal, 2003).

5.2 Results

Before presenting the final results, we first examine the effect of smoothing values for EM models. We test smoothing values from $\{1,10,20,30,40,50,60,70,80,90,100\}$ and evaluate the unlabeled attachment scores (UAS) of the basic model and the basic+bdr model on both the development set and the test set. Note the performance on the test set is only used for references, the final smoothing value is selected as the one with best performance on the development set. Experimental results are plotted in Figure 3. We can easily find that the best smoothing value (with highest dev-score) is 20 for both of these models. The basic+bdr model achieves significant (at $p < 10^{-3}$ level) better results (dev: 66.3, tst: 66.7) than the basic model (dev: 63.3, tst: 62.9) on both the development and test sets when the optimal smoothing values are selected.

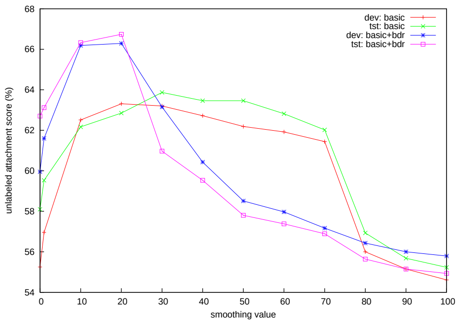


Figure 3: The effect of smoothing value on development and test set of PTB10

The final results of basic and basic+bdr models, using EM or Pitman-Yor process (PYP) are shown in Table 2 for comparison. Some existing results (copied from Figure 4(a) in (Bisk and Hockenmaier, 2012)) are also given in this table. Comparing within the four models described in this paper, we draw following conclusions:

1. The basic+bdr model achieves significant better results than the basic model under the EM learning. The boundary words capture lexical information about constituents and show complementary effectiveness for CCG induction.
2. The Bayesian framework outperforms the EM baseline significantly. This provides evidence that the compact models are preferred in the unsupervised CCG induction.
3. The combination of the boundary model and the Bayesian inference only show slightly better results than individual components. The reason might be that both the boundary words and the Bayesian model have the same effects and give high probabilities to those parse trees with more reused rules.

4. For longer sentences, the proposed methods still outperform baseline model with a large gap, demonstrating the robustness of our method.

Model		PTB10	PTB20
(Klein and Manning, 2004)		47.5	-
(Headden III et al., 2009)		68.8	-
(Spitkovsky et al., 2010)		65.3^*	53.8^*
(Cohn et al., 2010)		65.9	58.3
(Bisk and Hockenmaier, 2012)		71.5	60.3
(Naseem et al., 2010)		71.9	50.4^*
EM	basic	62.9	49.9
	basic+bdr	66.7^+	54.0^+
PYP	basic	66.0^+	53.9^+
	basic+bdr	66.7^+	55.1^+

Table 2: Comparison results on the test set with various length limits. Results of existing approaches are copied from (Bisk and Hockenmaier, 2012). Results with (*) were obtained with additional training data. Results with ($^+$) outperform the baseline (basic with EM) results significantly at $p < 10^{-3}$ level according to the McNemar's significant test.

Compared with existing approaches, our models stay in the intermediate level. Headden III et al. (2009) use rich contexts, words as well as POS tags, and sophisticated smoothing techniques, which might explain their higher performance than ours on short sentences. Naseem et al. (2010) manually specify some dependency rules in experiments, while we only use some coarse restrictions on the lexicon and grammar generation. Our models are mainly based on the previous work (Bisk and Hockenmaier, 2012). The full-EM model in (Bisk and Hockenmaier, 2012) corresponds to the basic model in our paper. Their reported results of full-EM are around 55−60 on short sentences, lower than our implementation. However, the best model in their paper outperforms our models on both short and long sentences. They achieve the state-of-the-art performance using the *k*-best EM learning. It should be emphasised that the *k*-best EM learning strategy is still applicable to our proposed basic+bdr model, which we leave for future work.

5.3 Discussion and Future work

Our method and many previous approaches (Klein and Manning, 2002, 2004; Bisk and Hockenmaier, 2012) take the golden part-of-speech tags as input. This practice may reduce data sparsity problem caused by directly modelling words. However, this may also lose useful lexical information. As reported in (Headden III et al., 2009), incorporating words with high frequencies (greater than 100 times in their experiments) as well as the POS tags could improve the induction accuracy for dependency models. In CCG, words may also help to distinguish lexical categories. For example, the transitive verbs are often tagged as `(S\N)/N` and the intransitive verbs often have category `S\N`. However, these syntactic differences are not encoded in the Penn treebank POS tags, in which they may both have the POS tag $\mathtt{VB}_x$ depending on the tenses. We leave this extension for future work.

Although the simple additive smoothing methods could improve EM results (see Figure 3), sophisticated smoothing schemes are also applicable (Headden III et al., 2009). Currently, the final probability is the product of basic CCG model and boundary model. This simple strategy already shows effectiveness in our experiments. In future work, different interpolation or back-off methods will be investigated. In addition, the context words have been proved useful for constituency tree induction (Klein and Manning, 2002; Golland et al., 2012). We could also integrate the context information to help CCG induction.

The performance of Bayesian model is somehow below our expectation, especially for the basic+bdr model. Currently, we simply use the grammars sampled at the last iteration to parse test sentences. Johnson and Goldwater (2009) propose the *maximum marginal decoding* technique to obtain more stable results, which we will explore in the future. Furthermore, we do not elaborately tune hyperparameters of Bayesian model, such as the prior distribution of $\boldsymbol{a}, \boldsymbol{b}$, the value of $\boldsymbol{\alpha}$, etc. Finally, tree nodes tend to be labeled with common and simple categories in CCGbank (Hockenmaier and Steedman, 2007). We could define probability models over the number of the categories and the internal arity of categories, and put sparse priors to enforce compact model.

6 Related work

Unsupervised dependency grammar induction has attracted a lot of research interests. The dependency model with valence (DMV) (Klein and Manning, 2004; Headden III et al., 2009; Cohen and Smith, 2009) is one of representative work, in which the valence is explicitly modelled. In contrast, the CCG formalism encodes the functor arity and word orders via syntactic categories, providing a more syntax-meaningful representation especially for long-range dependencies (Steedman, 2000). Since our work is based on CCG induction, we only present CCG-related work.

Osborne and Briscoe (1997) propose an unsupervised learning model for CCG induction. They create a labeled binary tree for each part-of-speech tag sequences in a greedy, bottom-up, incremental manner. The label of each inner node is the label of either the left or right sub-node. To avoid overfitting, they apply the Minimum Description Length (MDL) principle to learn compact grammars with minimal length of hypothesis and minimal length of data encoded in the hypothesis. While our model uses the alternative Bayesian learning method to learn compact grammars. Watkinson and Manandhar (1999) describe a CCG induction model based on linguistic lexicon generation. The learner is provided with a set of manually defined English oriented CCG categories, such as the verb-subcategorization. Compared to their work, we initialize lexicons with more general categories and learn complex categories and grammar rules automatically. Ponvert (2007) presents a generic algorithm to learn CCG categories. However, the reported experiments do not show much promising results. Naseem et al. (2010) use manually-specified linguistic-motivated rules in dependency grammar induction. Variational Bayesian method is used to estimate the parameters.

The most related work is (Bisk and Hockenmaier, 2012). The grammar generation step described in our paper is almost the same as the one in their paper. They compare various EM settings (full EM, Viterbi EM, and k-best EM) and find that the k-best EM could achieve best performance. They report the state-of-the-art results for unsupervised dependency grammar induction. Instead of the k-best EM, we perform the Bayesian inference and use sampling to estimate parameters. In addition, we exploit the use of rich lexical information and propose the boundary model to improve CCG induction. It is worth noting that the k-best EM can be also used for our boundary model, which we leave for future work.

Conclusion

In this paper, we have proposed to incorporate lexical information in the unsupervised CCG induction. Specifically, an additional boundary model is defined to capture complex language aspects, in which boundary words are generated from a special symbol independently for each span covered by tree nodes. Furthermore, we describe the nonparametric Pitman-Yor process to encourage rule reuse, resulting in compact grammars. Experimental results demonstrate that both the boundary model and the Bayesian inference outperform the baseline CCG induction system.

Acknowledgments

We would like to thank Zhixiang Ren for insightful discussions. We would also thank Professor Julia Hockenmaier for her LaTeX package to draw CCG derivations. Thank the anonymous reviewers for their helpful comments and suggestions.

References

Bisk, Y. and Hockenmaier, J. (2012). Simple robust grammar induction with combinatory categorial grammar. In *Proceedings of the Twenty-Sixth Conference on Artificial Intelligence (AAAI-12)*, pages 1643–1649, Toronto, Canada.

Blunsom, P. and Baldwin, T. (2006). Multilingual deep lexical acquisition for HPSGs via supertagging. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, pages 164–171, Sydney, Australia.

Blunsom, P. and Cohn, T. (2010). Unsupervised induction of tree substitution grammars for dependency parsing. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 1204–1213, Cambridge, MA.

Blunsom, P. and Osborne, M. (2008). Probabilistic inference for machine translation. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 215–223, Honolulu, Hawaii.

Bod, R. (2006). An all-subtrees approach to unsupervised parsing. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 865–872, Sydney, Australia.

Boonkwan, P. and Steedman, M. (2011). Grammar induction from text using small syntactic prototypes. In *Proceedings of 5th International Joint Conference on Natural Language Processing*, pages 438–446, Chiang Mai, Thailand.

Bos, J., Clark, S., Steedman, M., Curran, J. R., and Hockenmaier, J. (2004). Wide-coverage semantic representations from a CCG parser. In *Proceedings of Coling 2004*, pages 1240–1246, Geneva, Switzerland.

Clark, S. and Curran, J. (2003). Log-linear models for wide-coverage CCG parsing. In *Proceedings of the 2003 Conference on Empirical Methods in Natural Language Processing*, pages 97–104.

Clark, S. and Curran, J. R. (2007). Wide-coverage efficient statistical parsing with CCG and log-linear models. *Computational Linguistics*, 33(4):493–552.

Cohen, S. and Smith, N. A. (2009). Shared logistic normal distributions for soft parameter tying in unsupervised grammar induction. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 74–82, Boulder, Colorado.

Cohn, T., Blunsom, P., and Goldwater, S. (2010). Inducing Tree-Substitution grammars. *Journal of Machine Learning Research*, 11:3053–3096.

Cohn, T., Goldwater, S., and Blunsom, P. (2009). Inducing compact but accurate tree-substitution grammars. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 548–556, Boulder, Colorado.

Golland, D., DeNero, J., and Uszkoreit, J. (2012). A feature-rich constituent context model for grammar induction. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 17–22, Jeju Island, Korea.

Hassan, H., Sima'an, K., and Way, A. (2007). Supertagged phrase-based statistical machine translation. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 288–295, Prague, Czech Republic.

Hastings, W. K. (1970). Monte Carlo sampling methods using Markov chains and their applications. *Biometrika*, 57(1):97–109.

Headden III, W. P., Johnson, M., and McClosky, D. (2009). Improving unsupervised dependency parsing with richer contexts and smoothing. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 101–109, Boulder, Colorado.

Hockenmaier, J. and Steedman, M. (2002). Generative models for statistical parsing with combinatory categorial grammar. In *Proceedings of 40th Annual Meeting of the Association for Computational Linguistics*, pages 335–342, Philadelphia, Pennsylvania, USA.

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

Huang, Y., Zhang, M., and Tan, C. L. (2012). Improved constituent context model with features. In *Proceedings of the 26th Pacific Asia Conference on Language, Information and Computation (to appear)*, Bali, Indonesia.

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

Johnson, M. and Goldwater, S. (2009). Improving nonparameteric bayesian inference: experiments on unsupervised word segmentation with adaptor grammars. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 317–325, Boulder, Colorado.

Johnson, M., Griffiths, T. L., and Goldwater, S. (2007). Adaptor grammars: A framework for specifying compositional nonparametric bayesian models. In *Advances in Neural Information Processing Systems 19*, pages 641–648, Cambridge, MA.

Jones, B. K., Johnson, M., and Frank, M. C. (2010). Learning words and their meanings from unsegmented child-directed speech. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 501–509, Los Angeles, California.

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

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

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

Marcus, M. P., Marcinkiewicz, M. A., and Santorini, B. (1993). Building a large annotated corpus of English: the Penn Treebank. *Computational Linguistics*, 19(2):313–330.

Naseem, T., Chen, H., Barzilay, R., and Johnson, M. (2010). Using universal linguistic knowledge to guide grammar induction. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 1234–1244, Cambridge, MA.

Neal, R. M. (2003). Slice sampling. *Annals of Statistics*, 31(3):705–767.

Osborne, M. and Briscoe, T. (1997). Learning stochastic categorial grammars. In *Proceedings of CoNLL97: Computational Natural Language Learning*, pages 80–87.

Pitman, J. and Yor, M. (1997). The two-parameter Poisson-Dirichlet distribution derived from a stable subordinator. *Annals of Probability*, 25:855–900.

Ponvert, E. (2007). Inducing combinatory categorial grammars with genetic algorithms. In *Proceedings of the ACL 2007 Student Research Workshop*, pages 7–12, Prague, Czech Republic.

Seginer, Y. (2007). Fast unsupervised incremental parsing. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 384–391, Prague, Czech Republic.

Spitkovsky, V. I., Alshawi, H., Jurafsky, D., and Manning, C. D. (2010). Viterbi training improves unsupervised dependency parsing. In *Proceedings of the Fourteenth Conference on Computational Natural Language Learning*, pages 9–17, Uppsala, Sweden.

Steedman, M. (2000). *The Syntactic Process*. Cambridge, MA, USA.

Steedman, M. and Baldridge, J. (2011). Combinatory categorial grammar. In Borsley, R. and Börjars, K., editors, *Non-Transformational Syntax: Formal and Explicit Models of Grammar*, pages 181–224.

Teh, Y. W., Jordan, M. I., Beal, M. J., and Blei, D. M. (2006). Hierarchical Dirichlet processes. *Journal of the American Statistical Association*, 101(476):1566–1581.

van Zaanen, M. (2000). ABL: Alignment-based learning. In *Proceedings of the 18th International Conference on Computational Linguistics (Coling 2000)*, volume 2, pages 961–967, Saarbrücken, Germany.

Vijay-Shanker, K. and Weir, D. (1994). The equivalence of four extensions of context-free grammars. *Mathematical Systems Theory*, 27(6):511–546.

Watkinson, S. and Manandhar, S. (1999). Unsupervised lexical learning with categorial grammars using the LLL corpus. In *Proceedings of the 1st Workshop on Learning Language in Logic*.

Xiong, D., Zhang, M., and Li, H. (2010). Learning translation boundaries for phrase-based decoding. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 136–144, Los Angeles, California.

Zettlemoyer, L. and Collins, M. (2007). Online learning of relaxed CCG grammars for parsing to logical form. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 678–687, Prague, Czech Republic.

Zhang, Y. and Clark, S. (2011). Syntax-based grammaticality improvement using CCG and guided search. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 1147–1157, Edinburgh, Scotland, UK.