Two Models of Stochastic Dependency Grammar

Joakim Nivre

1 Introduction

One of the most prominent trends in research on natural language parsing during the last decade has been the development of data-driven methods and in particular the use of stochastic parsing models (see, e.g., Magerman [18], Bod [2, 3], Charniak [6, 7], Collins [8, 9, 10] and Goodman [13]). Another trend, although perhaps less prominent, has been a growing interest in models based on dependency grammar, exemplified by the collection of papers in Kahane and Polguère [17].

In this paper, I will be concerned with the combination of stochastic parsing and dependency grammar. More precisely, I will review two different attempts to develop stochastic models of dependency grammar, namely those of Carroll and Charniak [5] and Eisner [11, 12]. The central concepts of dependency grammar are introduced in section 2; the two proposals are discussed in sections 3–4; and conclusions are drawn in section 5.

2 Dependency Grammar

By and large, *dependency grammar* is a rather vague concept which is probably best understood as an umbrella term covering a large family of grammatical theories and formalisms that share certain basic assumptions about grammatical structure (see, e.g., Tesnière [25], Sgall, Hajicova and Panevova [23], Mel'cuk [19], Hudson [15, 16], Sleator and Temperley [24]). Foremost among these is the assumption that syntactic structure consists of *lexical nodes* linked by binary relations called *dependencies*. Thus, the common formal property of dependency structures, as compared to the more common syntactic representations based on constituency (or phrase structure), is the lack of phrasal nodes.

In a dependency structure, every lexical node except one is dependent on exactly one other lexical node, usually called its *head* or *regent*, which means that the structure can be represented as a connected graph, with nodes representing lexical elements and edges representing dependency relations. Normally we also require that the graph does not contain cycles, which means that it will in fact be a rooted tree with the root node representing the head of the sentence. Figure 1 illustrates the difference between a phrase structure tree (top) and a dependency graph (bottom) for the simple English sentence *she bought a car*:[1]

[1]There seems to be no general consensus as to whether the edges representing dependency relations should be drawn pointing from heads to dependents or vice versa (or indeed with arrowheads at all). In this paper I will adopt the former alternative, which seems most

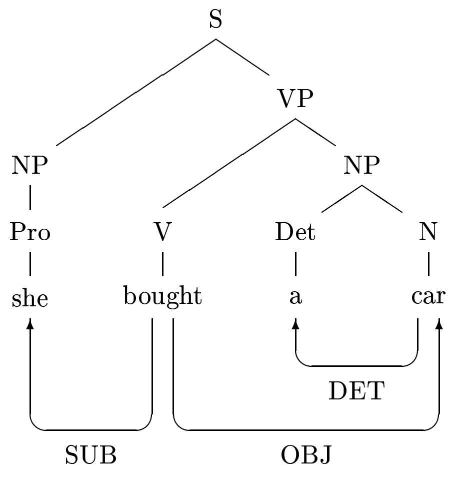


Figure 1: Comparison of parse tree (top) and dependency graph (bottom)

Given these basic assumptions about dependency structure, there are a number of parameters that can vary between different dependency grammars. For example, lexical nodes can be assumed to represent words, strings of words or even parts of words. Dependency relations can be directed or undirected, labeled or unlabeled.

Another set of issues concern the relation between dependency structure and word order (or surface realization generally). In many versions of dependency grammar, including the influential work of Tesnière [25], dependency structure is assumed to be independent of surface realization, which means that there is no linear order imposed on the nodes of the dependency graph. In other versions, including most dependency-based approaches to natural language parsing, the nodes of the dependency graph are ordered by a linear precedence relation representing the word order of the sentence. I will say that the latter kind of dependency graph is *linear* and the former *non-linear*. By extension, I will also talk about linear and non-linear dependency grammars.

The distinction between linear and non-linear dependency grammars is related to, but distinct from, the more well-known distinction between *projective* and *non-projective* dependency grammars. Projectivity (sometimes called *planarity*) is the requirement that a head and its dependents be realized in surface structure as a continuous string of words, where each individual node is also realized as a continuous string. In a linear dependency grammar, this amounts to the requirement that there are no crossing edges (which is the motivation for the term *planarity*).

There is a strong correlation between projectivity and linearity in that linear dependency grammars tend to assume projectivity, whereas non-projective grammars typically use non-linear dependency graphs (and handle surface realization in a separate component of the grammar). But there is no necessary connection, since linear dependency graphs with crossing edges can be used to

consistent with the assumption that the dependency graph forms a rooted tree.

represent non-projective surface realization, and non-linear dependency graphs can easily be combined with projective surface realization.

If there are many open issues with respect to the representation of dependency structure, it is even more unclear how dependency grammars should be defined as formal systems. First of all, much of the work on dependency grammar has been carried out in a descriptive linguistic tradition where formalization is not a central concern. But also within the more formally oriented computational linguistic community, there seems to be little consensus on the best way to formalize dependency grammars. Sometimes they are defined as a special case of context-free grammars, sometimes they are defined in a completely different framework. This diversity will be apparent also when we turn to the stochastic versions of dependency grammar in the following sections.

3 Stochastic Context-Free Dependency Grammar

According to Carroll and Charniak [5] a *dependency grammar* is a 3-tuple (S, V, R), where S is the start symbol, V is a vocabulary of terminal symbols (words), and R is a set of context-free rewrite rules.[2] The rules in R are restricted to the following two forms:

- $S \to \overline{w}$, where $w \in V$
- $\overline{w} \to \alpha w \beta$, where $w \in V$ and α, β are strings of zero or more $\overline{w'}$, for $w \in V$

In other words, Carroll and Charniak define dependency grammars as a special kind of context-free grammar $G = (N, V, R, S)$, where the non-terminal alphabet $N = \{S\} \cup \{\overline{w} \,|\, w \in V\}$, and where the productions in R are restricted to the following two forms:

- $S \to \overline{w}$, where $w \in V$
- $\overline{w} \to \alpha w \beta$, where $w \in V$ and $\alpha, \beta \in (N - \{S\})^*$

In the following I will say that a grammar satisfying these conditions is a *context-free dependency grammar* (CFDG).

The parse trees induced by a context-free dependency grammar are not dependency trees in the usual sense, since they contain nodes labeled with non-terminal symbols. However, it is straightforward to define a mapping from parse trees to dependency trees as follows:

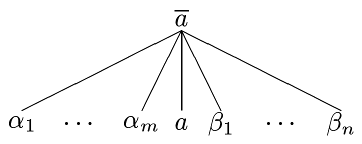
- Let π be a parse tree induced by a context-free dependency grammar G.
- The corresponding dependency tree $T = \delta(\pi)$, where δ is defined as follows:
    1. If x has the form

$$\begin{tikzcd} S \arrow[d, dash] \\ \alpha \end{tikzcd}$$

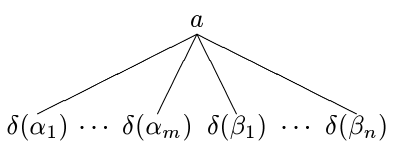
then $\delta(x) = \delta(\alpha)$

[2]Carroll and Charniak use the symbol N instead of V to denote the terminal vocabulary. In order to achieve consistency across sections, I have changed the symbols used by the original authors whenever necessary.

2. If x is a leaf node, then $\delta(x) = x$.
3. If x has the form



then $\delta(x) =$


a
$\delta(\alpha_1) \cdots \delta(\alpha_m) \; \delta(\beta_1) \cdots \delta(\beta_n)$

Under the normal interpretation of parse trees for context-free grammars, the parse trees induced by a CFDG are equivalent to *linear* dependency trees, since they encode the relative order of dependents both with respect to each other and with respect to the head. In order to maintain this property in the mapping outlined above, we need to distinguish between the *left* dependents $\delta(\alpha_1), \ldots, \delta(\alpha_m)$ and the *right* dependents $\delta(\beta_1), \ldots, \delta(\beta_n)$ of a given head w. Given these assumptions, a CFDG may be characterized as a *linear*, *projective* dependency grammar with *unlabeled* dependency relations (cf. section 2).

The class of languages $\mathcal{L}_{\text{CFD}}$ definable by a CFDG is a proper subset of the context-free languages $\mathcal{L}_{\text{CF}}$. To see that $\mathcal{L}_{\text{CFD}}$ is a subset of $\mathcal{L}_{\text{CF}}$, we only need to consider the fact that every CFDG is a context-free grammar (CFG). To see that is a *proper* subset, we consider the language $L_{abab} = \{abab\}$ over the terminal vocabulary $V = \{a, b\}$, which is clearly a context-free language but which cannot be generated by a CFDG.

- **Proposition:** There is no CFDG that generates L_{abab}.

- **Proof:** We prove the proposition by demonstrating that any CFDG that generates the string *abab* must generate an infinite language. Since L_{abab} is finite, this establishes the proposition. We begin by noting that if the string *abab* is generated by a CFDG G, then it must have a derivation in one of the following four forms:

  1. $S \Rightarrow \overline{a} \Rightarrow a\beta_1 \overset{*}{\Rightarrow} abab$
  2. $S \Rightarrow \overline{b} \Rightarrow \alpha_2 b \beta_2 \overset{*}{\Rightarrow} abab$
  3. $S \Rightarrow \overline{a} \Rightarrow \alpha_3 a \beta_3 \overset{*}{\Rightarrow} abab$
  4. $S \Rightarrow \overline{b} \Rightarrow \alpha_4 b \overset{*}{\Rightarrow} abab$

  where $\alpha_i, \beta_i \in (N - \{S\})^*$. Consider the first case. Since the terminal a can only be derived from the nonterminal $\overline{a}$, we must have that $\beta_1 \overset{*}{\Rightarrow} \alpha_{11}\overline{a}\beta_{11}$. But then we also have that $\overline{a} \overset{*}{\Rightarrow} a\alpha_{11}\overline{a}\beta_{11}$, which means that $\overline{a}$ is a recursive category and the grammar generates an infinite set of strings. The remaining three cases are analogous, with $\overline{b}$ being the necessarily recursive category in cases 2 and 4. Since one of the four cases has to hold for any CFDG that generates *abab*, the proposition follows.

Since the language L_{abab} is in fact a regular language, one may ask whether $\mathcal{L}_{\mathrm{CFD}}$ contains any languages that are not regular. To show that $\mathcal{L}_{\mathrm{CFD}}$ does contain non-regular languages, we show that there is a CFDG for the language $L_{a^nb^n} = \{a^nb^n \mid n \geq 1\}$ which is known not to be regular (Hopcroft et al. [14]).

- Let $G = (N, V, R, S)$ be the following CFDG:
  1. $N = \{S, \overline{a}, \overline{b}\}$
  2. $V = \{a, b\}$
  3. $R = \{S \rightarrow \overline{a}, \overline{a} \rightarrow a\overline{b}, \overline{b} \rightarrow \overline{a}b, \overline{b} \rightarrow b\}$
  4. $S = S$
- **Proposition:** $L(G) = L_{a^nb^n}$.
- **Proof:** We first prove that every string $\omega \in L_{a^nb^n}$ has a derivation in G by induction on the length of ω. If $|w| \leq 2$ then $\omega = ab$, which has the derivation $S \Rightarrow \overline{a} \Rightarrow a\overline{b} \Rightarrow ab$. If $|w| > 2$ then ω has the form $aw'b$, where $\omega' \in L_{abab}$ and $|w'| < |w|$. By the inductive hypothesis, $S \Rightarrow \overline{a} \overset{*}{\Rightarrow} w'$, which means that there must be a derivation for ω of the following form: $S \Rightarrow \overline{a} \Rightarrow a\overline{b} \Rightarrow a\overline{a}b \overset{*}{\Rightarrow} aw'b$. We then prove that every string $\omega \in L(G)$ is in $L_{a^nb^n}$ by induction on the number of steps in the derivation of ω. If ω is derived in at most 3 steps, then it must have the derivation $S \Rightarrow \overline{a} \Rightarrow a\overline{b} \Rightarrow ab$ and $ab \in L_{a^nb^n}$. If ω is derived in $n + 2$ steps (where $n \geq 2$) then the derivation must be of the form $S \Rightarrow \overline{a} \Rightarrow a\overline{b} \Rightarrow a\overline{a}b \overset{*}{\Rightarrow} aw'b$, where $\overline{a} \overset{*}{\Rightarrow} w'$ in $n - 1$ steps. But then there is also a derivation $S \overset{*}{\Rightarrow} w'$ in n steps, and since $\omega' \in L_{a^nb^n}$ (by the inductive hypothesis) it follows that $\omega \in L_{a^nb^n}$.

In sum, we have shown that the class of languages $\mathcal{L}_{\mathrm{CFD}}$ definable by context-free dependency grammars forms a proper subset of the context-free languages but includes languages that are not regular. To further characterize the class $\mathcal{L}_{\mathrm{CFD}}$ is a problem that we set aside for further research.

The *stochastic* dependency grammars considered by Carroll and Charniak [5] are simply the standard stochastic (or probabilistic) context-free grammars (Booth [4]) that result from restricting the underlying context-free grammars to be context-free dependency grammars in the sense defined above. Thus, a *stochastic context-free dependency grammar* (SCFDG) is a 5-tuple (N, V, R, S, P), where (N, V, R, S) is a CFDG and P is a function that assigns probabilities to the productions in R in such a way that the following conditions hold:

- For every nonterminal $n \in N$, $\sum_{\alpha} P(n \rightarrow \alpha) = 1$ (where α ranges over productions in R with n as their left hand side).
- Let ω be a string in $L(G)$, let $\{\pi_1, \ldots, \pi_m\}$ be the set of parse trees for ω according to G, and let $[r_{i1}, \ldots, r_{in_i}]$ be the bag of rules needed to construct the parse tree π_i in some canonical way. Then we have the following:
  1. $P(\pi_i) = \displaystyle\prod_{j=1}^{j=n_i} P(r_{ij})$

2. $P(\omega) = \sum_{i=1}^{i=m} P(\pi_i)$

In order to assign probabilities also to dependency trees, we extend the mapping from parse trees to dependency trees defined earlier as follows:

$$P(\delta(\pi_i)) = P(\pi_i)$$

Although the probabilistic model assumed in a SCFDG is the same as in a standard SCFG, the special form of the underlying CFDG in fact eliminates one of the weaknesses usually attributed to this model, viz. the lack of lexicalization. Since the nonterminal symbols of a CFDG (except for the start symbol) stand in a one-to-one correspondence with terminal symbols, conditioning rule probabilities on the left hand side becomes equivalent to conditioning on lexical heads. Moreover, since the nonterminal symbols occurring in the right hand side of the rule are also lexicalized, the probabilistic model of a PCFDG can in fact capture dependencies between a lexical head and its dependencies. This is illustrated in Figure 2, which shows a a simple SCFDG, and in Figure 3, which shows the parse tree π and dependency tree $T = \delta(\pi)$ assigned to the string *she bought a car* by the grammar. The probability of the analysis according to the grammar is $P(\pi) = P(T) = 0.125$.

S	$\rightarrow$	$\overline{\text{bought}}$	(1.0)
$\overline{\text{bought}}$	$\rightarrow$	$\overline{\text{she}}$ bought $\overline{\text{car}}$	(0.25)
$\overline{\text{bought}}$	$\rightarrow$	$\overline{\text{he}}$ bought $\overline{\text{car}}$	(0.25)
$\overline{\text{bought}}$	$\rightarrow$	$\overline{\text{she}}$ bought $\overline{\text{bike}}$	(0.25)
$\overline{\text{bought}}$	$\rightarrow$	$\overline{\text{he}}$ bought $\overline{\text{bike}}$	(0.25)
$\overline{\text{she}}$	$\rightarrow$	she	(1.0)
$\overline{\text{he}}$	$\rightarrow$	he	(1.0)
$\overline{\text{car}}$	$\rightarrow$	$\overline{\text{a}}$ car	(0.5)
$\overline{\text{car}}$	$\rightarrow$	$\overline{\text{the}}$ car	(0.5)
$\overline{\text{bike}}$	$\rightarrow$	$\overline{\text{a}}$ bike	(0.5)
$\overline{\text{bike}}$	$\rightarrow$	$\overline{\text{the}}$ bike	(0.5)
$\overline{\text{a}}$	$\rightarrow$	a	(1.0)
$\overline{\text{the}}$	$\rightarrow$	the	(1.0)

Figure 2: Stochastic CFDG

4 Weighted Bilexical Dependency Grammar

Eisner [11] presents three different probability models for dependency parsing, all of which can be viewed as instances of the general class of weighted bilexical dependency grammars defined in Eisner [12], although I will only discuss one of the models in this paper (model C). I will begin by characterizing the wider class of bilexical dependency grammars (without weights) and then move on to weighted grammars and in particular model C of Eisner [11].

A *bilexical dependency grammar* (BDG) consists of two elements:

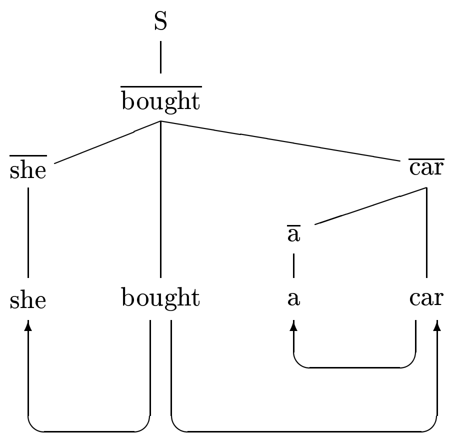


Figure 3: SCFDG parse tree π (top) and dependency tree $T = \delta(\pi)$ (bottom)

- A vocabulary V of terminal symbols (words), which contains a distinguished symbol ROOT.
- For each word $w \in V$, a pair of deterministic finite-state automata l_w and r_w. Each automaton accepts some regular subset of V^*.

The language $L(G)$ defined by a bilexical dependency grammar G is defined as follows:

- A *dependency tree* is a rooted tree whose nodes are labeled with words from V, and where the root node is labeled with the special symbol ROOT. The children of a node are ordered with respect to each other and the node itself, so that the node has both *left children* that precede it and *right children* that follow it.
- A dependency tree is *grammatical* according to G iff for every word token w that appears in the tree, l_w accepts the (possibly empty) sequence of w's left children (from right to left), and r_w accepts the sequence of w's right children (from left to right).
- A string $\omega \in V^*$ is generated by G with analysis T if T is a grammatical dependency tree according to G and listing the node labels of T in infix order yields the string ω followed by ROOT; ω is called the *yield* of T.
- The language $L(G)$ is the set of all strings generated by G.

Bilexical dependency grammars are similar in many respects to the context-free dependency grammars considered in section 3, both being linear, projective dependency grammars with unlabeled dependency relations. We have already seen that the class $\mathcal{L}_{\mathrm{CFD}}$ is a subset of the class $\mathcal{L}_{\mathrm{CF}}$ of context-free languages, and the same holds for the class $\mathcal{L}_{\mathrm{BD}}$ of languages definable by BDGs.

- **Proposition:** $\mathcal{L}_{BD} \subseteq \mathcal{L}_{CF}$

- **Proof:** We prove the proposition by showing that for every BDG G there is a CFG G' such that $L(G) = L(G')$. Let G be a BDG with vocabulary V and let $G' = (N, V - \{\textsc{root}\}, R, S)$ be a context-free grammar, where $N = \{n \mid n \to \alpha \in R\}$ and R is constructed as follows:

  1. For every $w \in V$, convert l_w and r_w to regular grammars with start symbols w_l and w_r, respectively, making sure that the nonterminal vocabularies of these regular grammars are mutually disjoint, and changing all terminal symbols w to $\overline{w}$. The productions of these regular grammars are in R.[3]
  2. For every $w \in V - \{\textsc{root}\}$, the production $\overline{w} \to w_l w w_r$ is in R.
  3. The production $S \to \textsc{root}_l$ is in R.[4]

  Then $L(G) = L(G')$, the proof of which we leave as the proverbial exercise to the reader.

However, the exact characterization of $\mathcal{L}_{BD}$ and its relation to $\mathcal{L}_{CFD}$ depends crucially on what restrictions (if any) we impose on the automaton l_{ROOT} defining the permissible (left) dependents of the special symbol $\textsc{root} \in V$. It seems in the spirit of dependency grammar to require that ROOT always has a single child,[5] but this requirement is not stated explicitly in Eisner [12]. Without the requirement, it seems that $\mathcal{L}_{\text{BD}}$ may well be a superset of $\mathcal{L}_{\text{CFD}}$, since we can then construct a BDG for the language L_{abab} shown in section 3 to be outside $\mathcal{L}_{\text{CFD}}$.[6] With the requirement, it is clear that $\mathcal{L}_{\text{BD}}$ fails to include some languages that are in $\mathcal{L}_{\text{CFD}}$, but the converse is also true.

- **Proposition:** $\mathcal{L}_{\text{CFD}} \not\subseteq \mathcal{L}_{\text{BD}}$
- **Proof:** Consider the language $L_{bab|cac} = \{bab, cac\}$ over the terminal vocabulary $V = \{a, b, c\}$.

  1. We first show that there is a CFDG G such that $L(G) = L_{bab|cac}$. Let $G = (N, V, R, S)$, where $N = \{S, \overline{a}, \overline{b}, \overline{c}\}$, $V = \{a, b, c\}$, $R = \{S \to \overline{a}, \overline{a} \to \overline{b}a\overline{b}, \overline{a} \to \overline{c}a\overline{c}, \overline{b} \to b, \overline{c} \to c\}$, and $S = S$. It is easy to prove that both bab and cac are generated by G and that any string generated by G is either bab or cac.
  2. We then show that there is no BDG G such that $L(G) = L_{bab|cac}$. We do this by proving that any BDG that generates the strings bab and cac must also generate other strings. Let G be a BDG that generates bab and cac. We assume first that the strings bab and cac are generated with a as the sole dependent of ROOT. Then it must be the case that the strings b and c are accepted by both l_a and r_a, which implies that bac and cab are in $L(G)$. On the other hand, if bab/cac can be generated with b/c as the sole dependent of the ROOT, then

[3]Methods for constructing a regular grammar from a finite automaton are described, for example, in Partee et al. [21] and Aho and Ullman [1].

[4]We may safely ignore $\textsc{root}_r$, since dependency trees with right children to the root are never grammatical.

[5]This restriction is in line with the requirement for CFDGs that productions with S as the left hand side are always of the form $S \to \overline{w}$.

[6]We simply let l_{ROOT} define L_{baba} (remember that left children are accepted right to left) and let all other automata define the empty language.

this b/c must dominate the other b/c in the dependency tree, which entails that $babab/cacac$ is also in $L(G)$. Hence, $L(G) \neq L_{bab|cac}$.

- **Proposition:** $\mathcal{L}_{\text{BD}} \not\subseteq \mathcal{L}_{\text{CFD}}$
- **Proof:** Consider the language $L_{(cb)^m a(bc)^n} = \{(cb)^m a(bc)^n \mid m, n \geq 0\}$ over the terminal vocabulary $V = \{a, b, c\}$.

  1. We first show that there is a BDG G such that $L(G) = L_{(cb)^m a(bc)^n}$. The automata of G are given in Figure 4, where $l_{\text{ROOT}} = (1)$, $l_a = r_a = (2)$, and $r_{\text{ROOT}} = l_b = r_b = l_c = r_c = (3)$. Since every grammatical dependency tree must have a as the sole dependent of ROOT, and since a can have zero or more occurrences of cb to the left (remember that left children are accepted right to left) and zero or more occurrences of bc to the left, it is clear that $L(G) = L_{(cb)^m a(bc)^n}$.
  2. We then show that there is no CFDG G such that $L(G) = L_{(cb)^m a(bc)^n}$. We do this by proving that any CFDG that generates strings of the form $(cb)^m a(bc)^n$ must also generate strings that are not of this form. Let G be a grammar that generates strings of the form $(cb)^m a(bc)^n$. Given the constraints on nonterminal symbols and productions in a CFDG, it must be the case that G contains the production $S \rightarrow \overline{a}$ and a production of the form $\overline{a} \rightarrow n_1 a n_2$, where $n_1, n_2 \in N$, $L(n_1) = (cb)^*$ and $L(n_2) = (bc)^*$. Both n_1 and n_2 can be either $\overline{b}$ or $\overline{c}$ (but nothing else), but in either case the grammar must contain the following productions:
     (a) $\overline{b} \rightarrow b\overline{c}$
     (b) $\overline{b} \rightarrow \overline{c}b$
     (c) $\overline{c} \rightarrow c\overline{b}$
     (d) $\overline{c} \rightarrow \overline{b}c$

     First of all, $\overline{b}$ and $\overline{c}$ has to be mutually recursive in order to allow arbitrarily long strings containing the same number of bs and cs. Secondly, both the left linear and the right linear productions are necessary in order to allow both sequences of cb and sequences of bc. But if G contains all four productions and n_1 and n_2 must be either $\overline{b}$ or $\overline{c}$, then G also generates strings of the form $(bc)^m a(bc)^n$ (and many other forms that should not be allowed). Hence, $L(G) \neq L_{(cb)^m a(bc)^n}$.

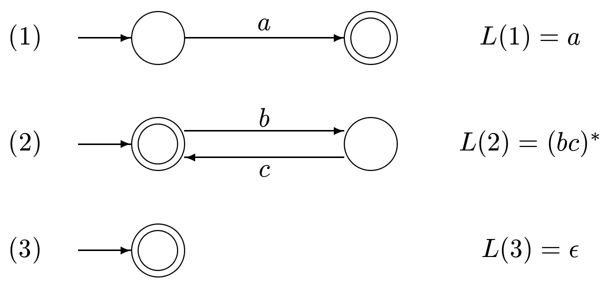


Figure 4: Automata for BDG G, $L(G) = L_{(cb)^m a(bc)^n}$

We turn now to weighted bilexical dependency grammars (WBDG), which are defined as follows in Eisner [12]:

- A *weighted DFA* A is a deterministic finite automaton that associates a real-valued *weight* with each arc and each final state (Mohri et al. [20]). Each accepting path through A is assigned a weight, namely the sum of all arc weights on the path and the weight of the final state. Each string ω accepted by A is assigned the weight of its accepting path.

- A *weighted bilexical dependency grammar* G is BDG in which all the automata l_w and r_w are weighted DFAs. The weight of a dependency tree T under G is defined as the sum, over all word tokens w in T, of the weight with which l_w accepts w's sequence of left children plus the weight with which r_w accepts w's sequence of right children.

Eisner [11] presents three different probabilistic models for dependency parsing, which can be reconstructed as different weighting schemes within the framework of bilexical dependency grammar. However, the first two models (models A and B) require that distinguish between an underlying string $\omega \in V^*$, described by the bilexical grammar, and a surface string Ω, which results from a possibly nondeterministic, possibly weighted finite-state transduction R on ω. The surface string Ω is then grammatical with analysis (T, P) if T is a grammatical dependency tree whose yield ω is transduced to Ω along an accepting path P in R. To avoid the distinction between underlying strings and surface strings, I will concentrate here on model C, which was found to perform significantly better than the other two models in the experiments reported in Eisner [11].

First of all, it should be pointed out that all the models discussed in Eisner [11] involve part-of-speech tags, in addition to word tokens and dependency relations, and define the joint probability of the words, tags and dependency links. Model C is defined as follows:

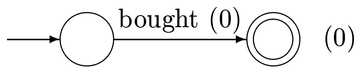
$$P(tw(1)\cdots tw(n), \text{links}) = \prod_{i=1}^{n} P(lc(i) \mid tw(i)) P(rc(i) \mid tw(i))$$

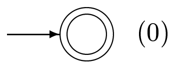
where $tw(i)$ is the ith tagged word, and $lc(i)$ and $rc(i)$ are the left and right children of the ith word, respectively. The probability of generating each child is conditioned on the tagged head word and the tag of the preceding child (left children being generated from right to left):

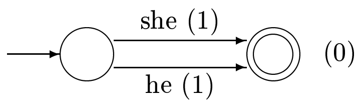
$$P(\text{lc}(i) \mid tw(i)) = \prod_{j=1}^{m} P(tw(lc_j(i)) \mid t(lc_{j-1}(i)), tw(i))$$
$$P(\text{rc}(i) \mid tw(i)) = \prod_{j=1}^{m} P(tw(rc_j(i)) \mid t(rc_{j-1}(i)), tw(i))$$

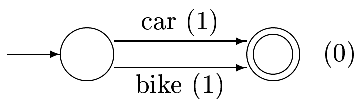
where $lc_j(i)$ is the jth left child of the ith word and $t(lc_{j-1}(i))$ is the tag of the preceding left child (and analogously $rc_j(i)$ and $t(rc_{j-1}(i))$ for right children). This model can be implemented in the WBDG framework by letting the automata l_w and r_w have weights on their arcs corresponding to the log of the probability of generating a particular left or right child given the tag of the preceding child. In this way, the weight assigned to a dependency tree T will be the log of $P(tw(1)\cdots tw(n), \text{links})$ as defined above. Figure 5 shows a simple WBDG (without part-of-speech tags) that generates the string *she bought a car* with the same dependency tree as in Figure 3 earlier and with the same probability ($W(T) = 3,\ P(T) = 0.125$).

$$V = \{\textsc{root}, \textit{bought}, \textit{she}, \textit{he}, \textit{car}, \textit{bike}, \textit{a}, \textit{the}\}$$

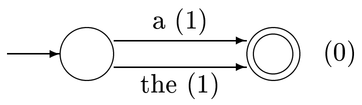
$l_{\text{ROOT}} =$ 

$r_{\text{ROOT}} =$ 

$l_{\text{bought}} =$ 

$r_{\text{bought}} =$ 

$$\begin{aligned}
l_{\text{she}} &= r_{\text{ROOT}} \\
r_{\text{she}} &= r_{\text{ROOT}} \\
l_{\text{he}} &= r_{\text{ROOT}} \\
r_{\text{he}} &= r_{\text{ROOT}}
\end{aligned}$$

$l_{\text{car}} =$ 

$$\begin{aligned}
r_{\text{car}} &= r_{\text{ROOT}} \\
l_{\text{bike}} &= l_{\text{car}} \\
r_{\text{bike}} &= r_{\text{ROOT}} \\
l_{\text{a}} &= r_{\text{ROOT}} \\
r_{\text{a}} &= r_{\text{ROOT}} \\
l_{\text{the}} &= r_{\text{ROOT}} \\
r_{\text{the}} &= r_{\text{ROOT}}
\end{aligned}$$

Figure 5: Weighted BDG

Comparing model C from Eisner [11] with the SCFDG model examined in section 3, we again find that the models are similar but not equivalent. Both models derive the probability of a dependency tree from the probability of dependents given their head, but Eisner's model assumes that left dependents are independent of right dependents and that the probability of the dependent sequence on each side of the head can be modeled by a Markov process. By contrast, in the SCFDG model each distinct configuration of dependents for a given head has its own probability, which means that a SCFDG will in general have more free parameters than the corresponding WBDG.

Thus, even in cases where it is possible to construct a WBDG which is equivalent to a given SCFDG both weakly and strongly, it may not be possible to construct an equivalent stochastic model. For example, consider the language $L_{(b|c)a(b|c)} = \{bab, bac, cab, cac\}$ over the terminal vocabulary $V = \{a, b, c\}$. Let G be a WBDG that generates the four strings of $L_{(b|c)a(b|c)}$ with a as the head in all cases, i.e., with the dependency trees depicted in Figure 6. Now, if $P(bab)$ is greater than both $P(bac)$ and $P(cab)$ according to G, then $P(bab)$ must also be greater than $P(cac)$. On the other hand, it is easy to construct a SCFDG that induces exactly the same dependency trees, but where it holds that $P(cac) > P(bab) > P(bac), P(cab)$, as is shown in Figure 7.

Of course, this does not necessarily mean that the SCFDG model is better suited for the analysis of natural language, since a larger number of free parameters also means that the model is more sensitive to sparse data. In fact, the expericence from treebank parsing seems to indicate that a Markov type model of the expansion of tree nodes is preferable, at least for traditional parse trees (Collins [9, 10] and Charniak [6, 7]).

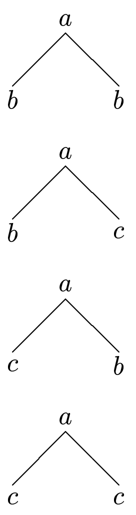


Figure 6: Dependency trees for $L_{(b|c)a(b|c)} = \{bab, bac, cab, cac\}$

$$\begin{array}{lll l}
S & \rightarrow & \overline{\mathrm{a}} & (1.0) \\
\overline{\mathrm{a}} & \rightarrow & \overline{\mathrm{b}}\ \mathrm{a}\ \overline{\mathrm{b}} & (0.3) \\
\overline{\mathrm{a}} & \rightarrow & \overline{\mathrm{b}}\ \mathrm{a}\ \overline{\mathrm{c}} & (0.15) \\
\overline{\mathrm{a}} & \rightarrow & \overline{\mathrm{c}}\ \mathrm{a}\ \overline{\mathrm{b}} & (0.15) \\
\overline{\mathrm{a}} & \rightarrow & \overline{\mathrm{c}}\ \mathrm{a}\ \overline{\mathrm{c}} & (0.4) \\
\overline{\mathrm{b}} & \rightarrow & \mathrm{b} & (1.0) \\
\overline{\mathrm{c}} & \rightarrow & \mathrm{c} & (1.0)
\end{array}$$

Figure 7: SCFDG for $L_{(b|c)a(b|c)} = \{bab, bac, cab, cac\}$

5 Conclusion

The two models of stochastic dependency grammar compared in this paper have many similarities, both being linear, projective dependency grammars with unlabeled dependency relations. In terms of expressive power, they both cover a proper subset of the context-free languages, although not the same subset.

Since the context-free dependency grammars of Carroll and Charniak [5] use productions that specify the left and right dependents of a head simultaneously, it is possible to capture constraints between dependents (as in the language $L_{bab|cac} = \{bab, cac\}$). On the other hand, in the bilexical dependency grammars of Eisner [12], the permissible sequences of left and right dependents are defined independently of each other, which means that it is possible to impose different constraints on these sequences even for arbitrarily long ones (as in the language $L_{(cb)^m a(bc)^n} = \{(cb)^m a(bc)^n \mid m, n \geq 0\}$). It is an interesting question which of these capacities are best suited to the description of natural languages.

Turning to the stochastic models, it should first be noted that whereas SCFDG is based on the standard SCFG model, the WBDG framework is in fact compatible with a variety of models, three of which are described in Eisner [11]. However, if we restrict the comparison to SCFDG and WBDG with Eisner's model C, we again find that there are great similarities in that both models capture dependencies between heads and their dependents by conditioning the latter on the former.

The main difference is again that SCFDG models each configuration of dependents separately, while WBDG instead treats left and right dependents independently of each other. Moreover, WBDG models the generation of dependents on either side of the head as a Markov process, where each new dependent is conditioned only on the part-of-speech of the preceding dependent (and on the head). The consequence of these differences is that SCFDG is less constrained in the sense that it can model a larger class of probability distributions for a given set of dependency trees. On the other hand, the WBDG model is more flexible and probably better suited for data-driven parsing, given its relatively smaller number of parameters.

Finally, let us note that neither of the two models studied in this paper can be said to exploit the full power of dependency grammar and stochastic modeling. First of all, they do not use the possibility of categorizing dependency relations, which allows a more fine-grained analysis of syntactic structure. Secondly, they are limited to projective dependency grammar, although it seems clear that certain constructions in natural language are non-projective in nature. Thirdly, the stochastic models are limited to relations between heads and their dependents, whereas a more sophisticated model could take more distant relations into account and make the relations between heads and dependents sensitive to the kind of dependency relation at hand (which in turn requires that dependency relations are categorized).

A very interesting proposal in this respect is the statistical theory of dependency syntax developed by Samuelsson [22], which provides a stochastic formalization of non-projective dependency grammar in terms of two separate stochastic processes, a top-down process generating dependency trees and a bottom-up process generating surface strings given dependency trees. To compare this theory with the models discussed in this paper is an interesting project for the future.

References

[1] Alfred V. Aho and Jeffrey D. Ullman. *Foundations of Computer Science.* Computer Science Press, 1995.

[2] Rens Bod. *Enriching Linguistics with Statistics: Performance Models of Natural Language.* PhD thesis, University of Amsterdam, 1995.

[3] Rens Bod. *Beyond Grammar.* CSLI Publications, 1998.

[4] T. L. Booth. Probabilistic representation of formal languages. In *IEEE Conference Record of the 1969 Tenth Annual Symposium on Switching and Automata Theory*, pages 77–81, 1969.

[5] John Carroll and Eugene Charniak. Two experiments on learning probabilistic dependency grammars from corpora. Technical Report TR-92, Department of Computer Science, Brown University, 1992.

[6] Eugene Charniak. Statistical parsing with a context-free grammar and word statistics. In *Proceedings of the Fourteenth National Conference on Artificial Intelligence.* MIT Press, 1997.

[7] Eugene Charniak. A maximum-entropy-inspired parser. In *Proceedings NAACL-2000*, 2000.

[8] Michael Collins. A new statistical parser based on bigram lexical dependencies. In *Proceedings of the 34th Annatual Meeting of the Association for Computational Linguistics*, pages 184–191, Santa Cruz, CA, 1996.

[9] Michael Collins. Three generative, lexicalised models for statistical parsing. In *Proceedings of the 35th Annatual Meeting of the Association for Computational Linguistics*, pages 16–23, Madrid, Spain, 1997.

[10] Michael Collins. *Head-Driven Statistical Models for Natural Language Parsing.* PhD thesis, University of Pennsylvania, 1999.

[11] Jason M. Eisner. Three new probabilistic models for dependency parsing: An exploration. In *Proceedings of COLING-96*, Copenhagen, 1996.

[12] Jason M. Eisner. Bilexical grammars and their cubic-time parsing algorithms. In Harry Bunt and Anton Nijholt, editors, *Advances in Probabilistic and Other Parsing Technologies.* Kluwer, 2000.

[13] Joshua Goodman. *Parsing Inside-Out.* PhD thesis, Harvard University, 1998.

[14] J. E. Hopcroft, R. Motwani, and J. D. Ullman. *Introduction to Automata Theory, Languages, and Computation.* Addison-Wesley, 2001.

[15] Richard A. Hudson. *Word Grammar.* Blackwell, 1984.

[16] Richard A. Hudson. *English Word Grammar.* Blackwell, 1991.

[17] Sylvain Kahane and Alain Polguère, editors. *Proceedings of the Workshop on Processing of Dependency-Based Grammars*, Université de Montréal, Quebec, Canada, August 1998.

[18] David M. Magerman. Statistical decision-tree models for parsing. In *Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics*, pages 276–283, Boston, MA, 1995.

[19] Igor Mel'cuk. *Dependency Syntax: Theory and Practice.* State University of New York Press, 1988.

[20] M. Mohri, F. Pereira, and M. (1996) Riley. Weighted automata in text and speech processing. In *Workshop on Extended Finite-State Models of Language (ECAI-96)*, pages 46–50, Budapest, 1996.

[21] Barbara H. Partee, Alice ter Meulen, and Robert E. Wall. *Mathematical Methods in Linguistics.* Kluwer, 1993.

[22] Christer Samuelsson. A statistical theory of dependency syntax. In *Proceedings COLING-2000.* Morgan Kaufmann, 2000.

[23] Petr Sgall, Eva Hajicova, and Jarmila Panevova. *The Meaning of the Sentence in Its Pragmatic Aspects.* Reidel, 1986.

[24] Daniel Sleator and Davy Temperley. Parsing english with a link grammar. Technical Report CMU-CS-91-196, Carnegie Mellon University, Computer Science, 1991.

[25] Lucien Tesnière. *Éléments de syntaxe structurale.* Editions Klincksieck, 1959.