Accurate Unbounded Dependency Recovery using Generalized Categorial Grammars

Luan NGUYEN[1] *Marten VAN SCHIJNDEL*[2] *William SCHULER*[2]
(1) University of Minnesota, Dept. of Computer Science and Engineering, Minneapolis, MN
(2) The Ohio State University, Dept. of Linguistics, Columbus, OH
lnguyen@cs.umn.edu, vanschm@ling.osu.edu, schuler@ling.osu.edu

ABSTRACT
Accurate recovery of predicate-argument dependencies is vital for interpretation tasks like information extraction and question answering, and unbounded dependencies may account for a significant portion of the dependencies in any given text. This paper describes a categorial grammar which, like other categorial grammars, imposes a small, uniform, and easily learnable set of semantic composition operations based on functor-argument relations, but like HPSG, is generalized to limit the number of categories used to those needed to enforce grammatical constraints. The paper also describes a novel reannotation system used to map existing resources based on Government and Binding Theory, like the Penn Treebank, into this categorial representation. This grammar is evaluated on an existing unbounded dependency recovery task (Rimell et al., 2009; Nivre et al., 2010).

KEYWORDS: Grammar and formalisms, Semantics.

1 Introduction

Accurate recovery of predicate-argument dependencies is vital for interpretation tasks like information extraction and question answering, and unbounded dependencies may account for a significant portion of the dependencies in any given text. Many current interpretation models are based on PCFGs, trained on syntactic annotations from the Penn Treebank (Marcus et al., 1993). These often recover dependencies as a post-process to parsing, and often are not able to retrieve unbounded dependencies if they are optimized on syntactic representations that leave these dependencies out.

Categorial grammars, on the other hand, have well-defined unbounded dependency representations based on functor-argument relations in a small and easily-learnable set of composition operations. Such grammars — in particular, Combinatory Categorial Grammar (CCG) (Steedman, 2000; Clark and Curran, 2007) — do well on unbounded dependency recovery tasks (Rimell et al., 2009) but not as well as models based on Head Driven Phrase-structure Grammar (HPSG) (Pollard and Sag, 1994; Miyao and Tsujii, 2005), given the same training. This may be attributed to implicit tradeoffs in many categorial frameworks that minimize the number of composition operations at the expense of large numbers of possible categories for each lexical item, which may lead to sparse data effects in training. HPSG models, in contrast, maintain a relatively large number of composition operations and a relatively small set of possible lexical categories, which are then used in a wider set of contexts.

Can categorial grammars, which have well-studied semantic representations and are well suited for interpretation, obtain better performance on a general unbounded dependency extraction task if it adopts an HPSG-like strategy of re-using types in various contexts? This paper describes a categorial grammar which, like HPSG, is generalized to limit the number of categories used to those needed to enforce grammatical constraints, but like other categorial grammars, imposes a small, uniform, and easily learnable set of semantic composition operations based on functor-argument relations. The paper also describes a novel reannotation system used to map existing resources based on Government and Binding Theory, like the Penn Treebank, into this categorial representation. This grammar is evaluated by training a state-of-the-art latent-variable parser on a version of the Penn Treebank reannotated into this generalized categorial grammar representation, and testing on an existing unbounded dependency recovery task (Rimell et al., 2009; Nivre et al., 2010).

The remainder of this paper is organized as follows. Section 2 describes related work on modeling unbounded dependencies and reannotation of syntax, Section 3 describes a generalized categorial grammar which provides transparent predicate-argument dependencies and generalizes similar categories across contexts, Section 4 describes how syntactically-annotated corpora can be reannotated into this framework, and Section 5 describes an evaluation of this system on an existing unbounded dependency recovery task.

2 Related Work

This section describes related work in unbounded dependencies and syntax reannotation.

2.1 Unbounded Dependencies

This paper describes a general-purpose reannotation system and its application to the specific task of reannotating local argument-passing treatments of long-distance dependencies in a generalized categorial grammar. Unbounded dependencies are dependencies between con-

stituents and points of attachment that have other constituents syntactically intervening. For example, the sentence *What does the First Amendment protect?* has a preposed constituent *what* that functions as a direct object of the transitive verb *protect*. The long distance between the source and destination of this type of dependency, paired with the relatively low probability of their occurrence in the language, and the fact that filler-gap annotations in syntactic resources such as the Penn Treebank are often stripped out, makes it very difficult for parsers to recognize this type of dependency correctly. While difficult to parse, this type of dependency is vital to the meaning of the sentence and of great importance in applications such as question answering and information extraction.

Rich grammar formalisms such as CCG (Steedman, 2000) have been shown to provide very accurate syntactic dependency extraction. This is thought to be because these formalisms explicitly model long-distance filler-gap dependencies inherent in relative clauses and interrogatives (Gildea and Hockenmaier, 2003). But formal systems like CCG, especially as reflected in reannotated Treebanks like CCGbank (Hockenmaier, 2003), comprise analyses of many phenomena: right node raising, light verbs, subject control, object control, coordinating conjunctions, etc., among which filler-gap constructions are just one component. Some of these analyses may be more easily automatically learnable (and thus more robust) than others. For example, CCG analyzes filler-gap constructions using essentially the same mechanism as right-side complementation, turning gaps into right-seeking functors using a special backward crossed composition combinator when gaps are not at the right periphery of constituents that contain them (Steedman, 2000). This distinction between peripheral and non-peripheral filler-gap constructions, and this merging of filler-gap constructions with right-side complementation, may produce different data densities in training corpora and make certain phenomena harder or easier to learn than, say, an HPSG-style analysis (Pollard and Sag, 1994), which treats all filler-gap constructions the same, but distinguishes constituents containing gaps from constituents awaiting complements.

Naturally, empirical questions about alternative syntactic analyses such as these cannot be resolved by fitting to syntactically annotated corpora, since the syntactic analyses do not agree. This paper therefore addresses the question of how to empirically distinguish between analyses of individual phenomena, or between entire formal systems, on the basis of learnability in a down-stream unbounded dependency recovery task. In particular, this paper describes a method for implementing syntactic analyses of various phenomena through automatic reannotation rules, which operate deterministically on a corpus like the Penn Treebank (Marcus et al., 1993) to produce a corpus with desired syntactic analyses. This reannotated corpus is then used to define a probabilistic grammar which is automatically annotated with additional latent variable values (Petrov and Klein, 2007) to introduce distinctions based on distributions of words and syntactic categories that increase the probability of the corpus (and improve the accuracy of parsing on held-out data), but do not affect the calculation of dependency structure. Dependency structures can then be extracted from parse trees produced by this grammar in a deterministic post-process, and mapped to the dependency representation used by Rimell et al. (2009) for evaluation.

2.2 Reannotation

The reannotation process described in this paper is similar in purpose to efforts to convert Penn Treebank annotations into other grammar formalisms, e.g. CCG (Hockenmaier, 2003) and HPSG (Miyao et al., 2004). These reannotation processes consist of multiple phases for

head-finding, binarization, and relabeling. However, unlike the Hockenmaier or Miyao et al. processes, the rule set described in this paper is intended to be modified and reused by researchers interested in manipulating analyses of individual phenomena (like distinguishing or consolidating filler-gap and right-node raising constructions) to test the generality and learnability of alternative syntactic representations on some down-stream task such as recovering unbounded dependencies. As such, the reannotation rules used in this process are defined to apply in a single top-down pass, pulling arguments and modifiers out of constituents until only a lexical head remains at the bottom. This single-pass architecture allows the rules for reannotating various linguistic phenomena to be relatively modular, so that they can be independently manipulated and evaluated.

The reannotation rules described in this paper are also similar to reannotation rules in the 'wsjsed' system (Blaheta, 2002), but that system was evaluated as an error-correction tool, rather than for making theoretically-motivated changes to syntactic analyses.

The reannotated grammar described in this paper exploits the rich traces and function labels in the Treebank that are typically ignored in parsing. These annotations are transformed into a categorial grammar representation of filler-gap constructions that makes use of 'gap' arguments, similar to the 'slash' arguments used in HPSG. Other approaches also exploit the trace and function labels of the Penn Treebank. Campbell (2004) proposed recovering trace information in a post-process following parsing. In contrast, the sample reannotation described in this paper incorporates filler-gap annotations into the corpus on which the parser is trained. Collins (1997), connected fillers with gaps and introduced a 'TRACE' category into the training data. Since this annotation does not change the constituent structure of the corpus, the addition of this gap annotation did not result in improved parsing results for Collins' gap-annotating model (model 3) as compared to the non-gap-annotating model (model 2). However, these are the kind of unbounded dependencies that need to be recovered in the evaluation described in this paper.

3 Generalized Categorial Grammar

This paper explores the use of a generalized categorial grammar which has the transparent predicate-argument dependencies of traditional categorial grammars (based on function application), but is generalized to allow arbitrary sets of type-constructing operators. An extended set of type-constructing operators and a corresponding set of inference rules are then used to group syntactically-interchangeable signs — for example, those with peripheral and non-peripheral gaps or those occurring in post-nominal and predicative contexts — into equivalent categories.

A generalized categorial grammar (Lambek, 1958; Bach, 1981; Oehrle, 1994) is a tuple $\langle U, O, R, X, M \rangle$ of a set U of primitive category types, a set O of type-constructing operators, a set R of inference rules, a set X of vocabulary items, and a mapping M from vocabulary items to complex types. The set of primitive category types U specify various linguistic forms for descriptions of entities or eventualities, corresponding to different clause types,[1] e.g.:

[1] This system of categories may be viewed as a simplification of a tectogrammatical type system such as that of Mihalicek and Pollard (2010), weakened to be representable as a context-free grammar.

V: finite verbal (*they knew it*)
I: infinitive verbal (*them to know it*)
B: base-form verbal (*them know it*)
L: participial verbal (*them known it*)
A: adjectival/predicative (*them knowing it*)
R: adverbial (*them knowingly*)
G: gerund (*them knowing it*)
N: nominal form (e.g. *their knowledge of it*)
D: determiner (*their knowledge of it's*)
O: genitive (*of their knowledge of it*)
E: embedded infinitive (*for them to know it*)
C: complementized finite (*that they know it*)
Q: interrogative (*did they know it*)
S: complete utterance (*know it*)

The set of type-constructing operators O specify various kinds of arguments:[2]

-a: initial argument
-b: final argument
-c: initial conjunct
-d: final conjunct
-g: filler-gap argument
-h: held argument for right node raising
-i: interrogative pronoun argument
-r: relative pronoun argument

Using this set of primitive category types U and type-constructing operators O, a set of complex categories C can be defined such that:

1. every U is in C
2. every $C \times O \times C$ is in C
3. nothing else is in C

Mapping M defines associations from vocabulary items $x \in X$ to meaning functions and associated categories of the form '$(\lambda ...) : u\varphi_1...\varphi_v\psi$', where '$(\lambda ...)$' is a meaning function and '$u\varphi_1...\varphi_v\psi$' is a category consisting of output category $u \in U$, a sequence of argument categories $\varphi_1, ..., \varphi_v \in \{\textbf{-a}, \textbf{-b}, \textbf{-c}, \textbf{-d}\} \times C$, and an optional non-local argument category $\psi \in (\{\textbf{-r}, \textbf{-i}\} \times C) \cup \{\epsilon\}$. Since this model will be used to generate predicate-argument relations but not scoping relations, these meaning functions are constrained to describe simple existentially-quantified variables over instances of entities or eventualities, connected by a set of numbered argument relations. These meaning functions map instances of entities or eventualities i, j, k to truth values based on whether the described argument relations hold between these referents. These argument relations are defined as numbered functions $(v\, i){=}j$ from eventuality or predicate instances i to argument instances j identified by the number of the function v. The '0' function identifies j as i's predicate concept (so '0' maps entity or eventuality instances to instances of concepts associated with words in X), the '1' function identifies j as i's first argument (e.g. its subject), the '2' function identifies j as i's second argument (e.g. its direct object), and so on.[3] A graphical representation of the predicate-argument relations generated by this system for the sentence *The person who officials say stole millions,* is shown in Figure 1. This is similar to the semantic dependency representations of Mel'čuk (1988) and Parsons (1990).

[2] The **-a** and **-b** operators may be viewed as equivalent to the forward and backward slash, respectively, of Lambek (1958) or Bar-Hillel (1953) categorial grammars, except that they are not used to represent gap arguments or conjuncts. The **-g** operator is similar to the vertical or neutral slash of Kubota and Levine (2012), used to represent gap arguments. The **-c** and **-d** operators for conjuncts, the **-h** operator for rightward raising, and the **-r** and **-i** operators for relative and interrogative pronoun referents are novel extensions to the system.

[3] More sophisticated meaning functions are possible, but are not necessary for evaluating the accuracy of unbounded dependency recovery.

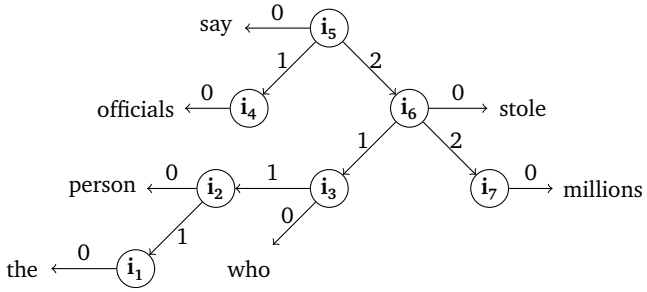


Figure 1: Graphical representation of predicate-argument dependencies for the sentence *The person who officials say stole millions.*

The meaning functions associated with most words specify just the predicate concept (which is here defined to match the word x):

$$x \;\mapsto_M\; (\lambda_i\,(0\,i){=}x) : u\varphi_1 \ldots \varphi_v \tag{1a}$$

Meaning functions for relative pronouns (Equation 1b) and interrogative pronouns (Equation 1c) introduce additional arguments k, using operators **-r** or **-i** for the referent of the antecedent of a relative or interrogative pronoun respectively:

$$x \;\mapsto_M\; (\lambda_{k\,i}\,(0\,i){=}x \wedge (v\,i){=}k) : u\varphi_1 \ldots \varphi_{v-1}\textbf{-r}c \tag{1b}$$

$$x \;\mapsto_M\; (\lambda_{k\,i}\,(0\,i){=}x \wedge (v\,i){=}k) : u\varphi_1 \ldots \varphi_{v-1}\textbf{-i}c \tag{1c}$$

Inference rules are defined in terms of *composition functions* for arguments, modifiers, and conjuncts. These composition functions each take a meaning function g for an initial (left) child sign and a meaning function h for a final (right) child sign (each defining a set of entity or eventuality instances) and return a meaning function for the parent, which is itself a function from entity or eventuality instances i to truth values:

- Composition functions for arguments $f_{u\varphi_1 \ldots \varphi_v}$ connect the referent j of an initial (left) child function g as an argument of referent i of a final (right) child function h, or vice versa:

$$f_{u\varphi_1 \ldots \varphi_{v-1}\textbf{-a}c} \stackrel{\text{def}}{=} \lambda_{g\,h\,i}\,\exists_j\,(v\,i){=}j \wedge (g\,j) \wedge (h\,i) \tag{2a}$$

$$f_{u\varphi_1 \ldots \varphi_{v-1}\textbf{-b}c} \stackrel{\text{def}}{=} \lambda_{g\,h\,i}\,\exists_j\,(v\,i){=}j \wedge (g\,i) \wedge (h\,j) \tag{2b}$$

- Composition functions for initial and final modifiers (f_{IM} and f_{FM}) are category-independent and return the referent of the argument (j) rather than of the predicate (i):

$$f_{\text{IM}} \stackrel{\text{def}}{=} \lambda_{g\,h\,j}\,\exists_i\,(1\,i){=}j \wedge (g\,i) \wedge (h\,j) \tag{3a}$$

$$f_{\text{FM}} \stackrel{\text{def}}{=} \lambda_{g\,h\,j}\,\exists_i\,(1\,i){=}j \wedge (g\,j) \wedge (h\,i) \tag{3b}$$

- Composition functions for conjuncts are similar to composition functions for arguments, except that they only count conjunct arguments, for $c, d \in C$:[4]

$$f_{c\text{-}\mathbf{c}d} \stackrel{\text{def}}{=} \lambda_{ghi}\, \exists_j\, (1\, i){=}j \wedge (g\, j) \wedge (h\, i) \tag{4a}$$

$$f_{c\text{-}\mathbf{d}d} \stackrel{\text{def}}{=} \lambda_{ghi}\, \exists_j\, (2\, i){=}j \wedge (g\, i) \wedge (h\, j) \tag{4b}$$

$$f_{\&} \stackrel{\text{def}}{=} \lambda_{ghi}\, \exists_{jk}\, j{=}(2\, i) \wedge (0\, i){=}(0\, j) \wedge (1\, j){=}k \wedge (g\, k) \wedge (h\, j) \tag{4c}$$

The set of *inference rules* R in the categorial grammar then apply these composition functions to compose and categorize super-lexical signs. These inference rules will use variables f, g, h over meaning functions, variables k over referents for possible values of gaps, variables $u \in U$ over primitive categories, variables $c, d, e \in U \times (\{\textbf{-a}, \textbf{-b}, \textbf{-c}, \textbf{-d}\} \times C)^*$ over categories with local arguments, and variables $\psi \in \{\textbf{-g}, \textbf{-h}, \textbf{-i}, \textbf{-r}\} \times C$ over non-local operators and argument categories:[5]

1. Inference rules for argument attachment apply functors of category c-**a**d or c-**b**d to initial or final arguments of category d. Non-local arguments k, using non-local operator and argument category ψ, are then propagated to the consequent from all possible combinations of antecedents, skipping over the composition function:

$$\frac{g{:}d \quad h{:}c\text{-}\mathbf{a}d}{(f_{c\text{-}\mathbf{a}d}\, g\, h){:}\, c} \quad \frac{g{:}d\psi \quad h{:}c\text{-}\mathbf{a}d}{\lambda_k\, (f_{c\text{-}\mathbf{a}d}\, (g\, k)\, h){:}\, c\psi} \quad \frac{g{:}d \quad h{:}c\text{-}\mathbf{a}d\psi}{\lambda_k\, (f_{c\text{-}\mathbf{a}d}\, g\, (h\, k)){:}\, c\psi} \quad \frac{g{:}d\psi \quad h{:}c\text{-}\mathbf{a}d\psi}{\lambda_k\, (f_{c\text{-}\mathbf{a}d}\, (g\, k)\, (h\, k)){:}\, c\psi} \quad \text{(Aa–d)}$$

$$\frac{g{:}c\text{-}\mathbf{b}d \quad h{:}d}{(f_{c\text{-}\mathbf{b}d}\, g\, h){:}\, c} \quad \frac{g{:}c\text{-}\mathbf{b}d\psi \quad h{:}d}{\lambda_k\, (f_{c\text{-}\mathbf{b}d}\, (g\, k)\, h){:}\, c\psi} \quad \frac{g{:}c\text{-}\mathbf{b}d \quad h{:}d\psi}{\lambda_k\, (f_{c\text{-}\mathbf{b}d}\, g\, (h\, k)){:}\, c\psi} \quad \frac{g{:}c\text{-}\mathbf{b}d\psi \quad h{:}d\psi}{\lambda_k\, (f_{c\text{-}\mathbf{b}d}\, (g\, k)\, (h\, k)){:}\, c\psi} \quad \text{(Ae–h)}$$

   For example, to attach a verb to a direct object with or without a gap:

$$\frac{\underline{\text{read}} \quad \underline{\text{a book about cars}}}{\mathbf{V\text{-}aN\text{-}bN} \qquad \mathbf{N}}\ \text{Ae} \qquad \frac{\underline{\text{read}} \quad \underline{\text{a book about}}}{\mathbf{V\text{-}aN\text{-}bN} \qquad \mathbf{N\text{-}gN}}\ \text{Ag}$$
$$\mathbf{V\text{-}aN} \qquad\qquad \mathbf{V\text{-}aN\text{-}gN}$$

2. Inference rules for modifier attachment apply initial or final modifiers of category u-**a**d to modificands of category c (again propagating non-local arguments ψ to the consequent from all combinations of antecedents, so as to skip over the composition function):

$$\frac{g{:}u\text{-}\mathbf{a}d \quad h{:}c}{(f_{\text{IM}}\, g\, h){:}c} \quad \frac{g{:}u\text{-}\mathbf{a}d\psi \quad h{:}c}{\lambda_k\, (f_{\text{IM}}\, (g\, k)\, h){:}c\psi} \quad \frac{g{:}u\text{-}\mathbf{a}d \quad h{:}c\psi}{\lambda_k\, (f_{\text{IM}}\, g\, (h\, k)){:}c\psi} \quad \frac{g{:}u\text{-}\mathbf{a}d\psi \quad h{:}c\psi}{\lambda_k\, (f_{\text{IM}}\, (g\, k)\, (h\, k)){:}c\psi} \quad \text{(Ma–d)}$$

$$\frac{g{:}c \quad h{:}u\text{-}\mathbf{a}d}{(f_{\text{FM}}\, g\, h){:}c} \quad \frac{g{:}c\psi \quad h{:}u\text{-}\mathbf{a}d}{\lambda_k\, (f_{\text{FM}}\, (g\, k)\, h){:}c\psi} \quad \frac{g{:}c \quad h{:}u\text{-}\mathbf{a}d\psi}{\lambda_k\, (f_{\text{FM}}\, g\, (h\, k)){:}c\psi} \quad \frac{g{:}c\psi \quad h{:}u\text{-}\mathbf{a}d\psi}{\lambda_k\, (f_{\text{FM}}\, (g\, k)\, (h\, k)){:}c\psi} \quad \text{(Me–h)}$$

   For example, to attach an adverbial modifier with or without a gap:

$$\frac{\underline{\text{sleep}} \quad \underline{\text{in Aix}}}{\mathbf{V\text{-}aN} \qquad \mathbf{R\text{-}aN}}\ \text{Me} \qquad \frac{\underline{\text{sleep}} \quad \underline{\text{in}}}{\mathbf{V\text{-}aN} \qquad \mathbf{R\text{-}aN\text{-}gN}}\ \text{Mg}$$
$$\mathbf{V\text{-}aN} \qquad\qquad \mathbf{V\text{-}aN\text{-}gN}$$

[4] The last of these (4c) introduces lexical and compositional relations for elided conjunctions in sequences of three or more conjuncts (e.g. between *creditors* and *investors* in the conjunction *creditors, investors, and employees*).

[5] A deductive system consists of inference rules of the form $\frac{P}{Q}R$, meaning premises or antecedents P entail conclusion or consequent Q according to rule or side condition R (Shieber et al., 1995). Additionally, this notation assumes adjacent premises arise from adjacent and similarly ordered sequences of observations.

3. Inference rules for conjunct attachment apply conjunctions of category c-**c**d or c-**d**d to conjuncts of category d (including repeated initial conjuncts):

$$\frac{g{:}d \quad h{:}\,c\text{-}\mathbf{c}d}{(f_{c\text{-}\mathbf{c}d}\ g\ h){:}\,c} \qquad \frac{g{:}d \quad h{:}\,c\text{-}\mathbf{c}d}{(f_{\&}\ g\ h){:}\,c\text{-}\mathbf{c}d} \qquad \frac{g{:}\,c\text{-}\mathbf{d}d \quad h{:}d}{(f_{c\text{-}\mathbf{d}d}\ g\ h){:}\,c} \tag{Ca–c}$$

$$\frac{g{:}d\psi \quad h{:}\,c\text{-}\mathbf{c}d\psi}{\lambda_k(f_{c\text{-}\mathbf{c}d}\ (g\ k)\ (h\ k)){:}\,c} \qquad \frac{g{:}d\psi \quad h{:}\,c\text{-}\mathbf{c}d\psi}{\lambda_k(f_{\&}\ (g\ k)\ (h\ k)){:}\,c\text{-}\mathbf{c}d\psi} \qquad \frac{g{:}\,c\text{-}\mathbf{d}d\psi \quad h{:}d\psi}{\lambda_k(f_{c\text{-}\mathbf{d}d}\ g\ (h\ k)){:}\,c} \tag{Cd–f}$$

For example, to combine three noun phrase conjuncts:

$$\dfrac{\dfrac{\text{creditors}}{\mathbf{N}} \quad \dfrac{\dfrac{\text{investors}}{\mathbf{N}} \quad \dfrac{\dfrac{\text{and}}{\mathbf{N\text{-}cN\text{-}dN}} \quad \dfrac{\text{employees}}{\mathbf{N}}}{\mathbf{N\text{-}cN}}\ \text{Cc}}{\mathbf{N\text{-}cN}}\ \text{Cb}}{\mathbf{N}}\ \text{Ca}$$

4. Inference rules for gap attachment hypothesize gaps as initial arguments, final arguments, or modifiers:[6]

$$\frac{g{:}\,c\text{-}\mathbf{a}d}{\lambda_k\,(f_{c\text{-}\mathbf{a}d}\ \{k\}\ g){:}\,c\text{-}\mathbf{g}d} \qquad \frac{g{:}\,c\text{-}\mathbf{b}d}{\lambda_k\,(f_{c\text{-}\mathbf{b}d}\ g\ \{k\}){:}\,c\text{-}\mathbf{g}d} \qquad \frac{g{:}c}{\lambda_k\,(f_{\mathrm{IM}}\ \{k\}\ g){:}c\text{-}\mathbf{g}d} \tag{Ga–c}$$

For example:

$$\dfrac{\text{is sleeping}}{\dfrac{\mathbf{V\text{-}aN}}{\mathbf{V\text{-}gN}}\ \text{Ga}} \qquad \dfrac{\text{we} \quad \dfrac{\text{drove}}{\mathbf{V\text{-}aN\text{-}bN}}}{\dfrac{\mathbf{N}}{\mathbf{V\text{-}gN}}\ \text{Ac}}\ \text{Gb} \qquad \dfrac{\text{is sleeping}}{\dfrac{\mathbf{V\text{-}aN}}{\mathbf{V\text{-}aN\text{-}gR}}\ \text{Gc}}$$

5. Inference rules for filler attachment apply gapped clauses to modificands or relative or interrogative phrases as fillers:

$$\frac{g{:}e \quad h{:}\,c\text{-}\mathbf{g}d}{\lambda_i\,\exists_j\,(g\ i)\wedge(h\ i\ j){:}e} \qquad \frac{g{:}d\text{-}\mathbf{r}e \quad h{:}\,c\text{-}\mathbf{g}d}{\lambda_{k\,j}\,\exists_i\,(g\ k\ i)\wedge(h\ i\ j){:}\,c\text{-}\mathbf{r}e} \qquad \frac{g{:}d\text{-}\mathbf{i}e \quad h{:}\,c\text{-}\mathbf{g}d}{\lambda_{k\,j}\,\exists_i\,(g\ k\ i)\wedge(h\ i\ j){:}\,c\text{-}\mathbf{i}e} \tag{Fa–c}$$

For example:

$$\dfrac{\dfrac{\text{the car}}{\mathbf{N}} \quad \dfrac{\text{we drove}}{\mathbf{V\text{-}gN}}}{\mathbf{N}}\ \text{Fa} \qquad \dfrac{\dfrac{\text{which}}{\mathbf{N\text{-}rN}} \quad \dfrac{\text{we drove}}{\mathbf{V\text{-}gN}}}{\mathbf{V\text{-}rN}}\ \text{Fb} \qquad \dfrac{\dfrac{\text{what}}{\mathbf{N\text{-}iN}} \quad \dfrac{\text{do we drive}}{\mathbf{Q\text{-}gN}}}{\mathbf{Q\text{-}iN}}\ \text{Fc}$$

6. Inference rules for relative pronoun attachment apply pronominal relative clauses of category c-**r**d to modificands of category e:

$$\frac{g{:}e \quad h{:}c\text{-}\mathbf{r}d}{\lambda_i\,\exists_j\,(g\ i)\wedge(h\ i\ j){:}e} \tag{R}$$

For example:

$$\dfrac{\dfrac{\text{the car}}{\mathbf{N}} \quad \dfrac{\text{which we drove}}{\mathbf{V\text{-}rN}}}{\mathbf{N}}\ \text{R}$$

[6] Here, set notation is used in order to save space: $\{k\} = (\lambda_i\ i{=}k)$.

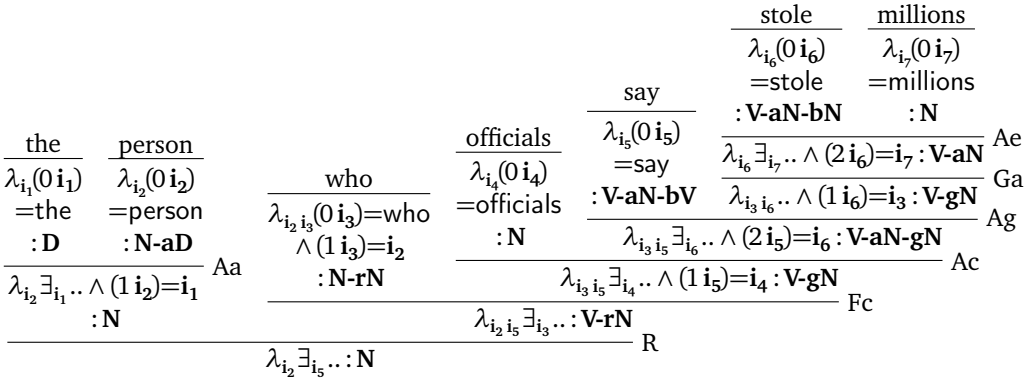
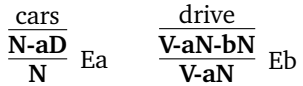


Figure 2: Example categorization of the noun phrase *the person who officials say stole millions*. This derivation yields the following lexical relations: $(0\,\mathbf{i_1})$=the, $(0\,\mathbf{i_2})$=person, $(0\,\mathbf{i_3})$=who, $(0\,\mathbf{i_4})$=officials, $(0\,\mathbf{i_5})$=say, $(0\,\mathbf{i_6})$=stole, $(0\,\mathbf{i_7})$=millions, and the following argument relations: $(1\,\mathbf{i_2})=\mathbf{i_1}$, $(1\,\mathbf{i_3})=\mathbf{i_2}$, $(1\,\mathbf{i_5})=\mathbf{i_4}$, $(2\,\mathbf{i_5})=\mathbf{i_6}$, $(1\,\mathbf{i_6})=\mathbf{i_3}$, $(2\,\mathbf{i_6})=\mathbf{i_7}$. The semantic dependency relations for this sentence are represented graphically in Figure 1.

7. Inference rules for argument elision (including determiners of plural nouns) simply leave these arguments unspecified in the resulting meaning function:

$$\frac{g:c\text{-}\mathbf{a}d}{g:c} \qquad \frac{g:c\text{-}\mathbf{b}d}{g:c} \qquad \frac{g:c\text{-}\mathbf{a}d\psi}{g:c\psi} \qquad \frac{g:c\text{-}\mathbf{b}d\psi}{g:c\psi} \qquad \text{(Ea–d)}$$

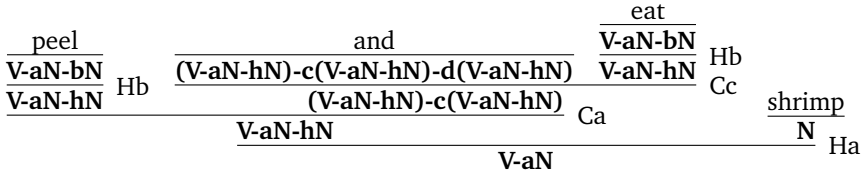
   For example, to elide determiners of plural nouns or optional direct objects:



8. Inference rules for right node raising introduction and attachment treat right-node raising as a type of non-local argument using operator **-h**:

$$\frac{g:c\text{-}\mathbf{h}d \quad h:d}{\lambda_i\,\exists_j\,(g\;j\;i)\wedge(h\;j):c} \qquad \frac{g:c\text{-}\mathbf{b}d}{\lambda_k\,(f_{c\text{-}\mathbf{b}d}\;g\;\{k\}):c\text{-}\mathbf{h}d} \qquad \text{(Ha–b)}$$

   For example:



An example derivation of the noun phrase *the person who officials say stole millions*, exemplifying F, G, and R rules, is shown in Figure 2; and an example derivation of the noun phrase *creditors, investors and employees of the company*, exemplifying C, E, and H rules, is shown in Figure 3. After all lambda expressions are applied to arguments in a derivation, each word

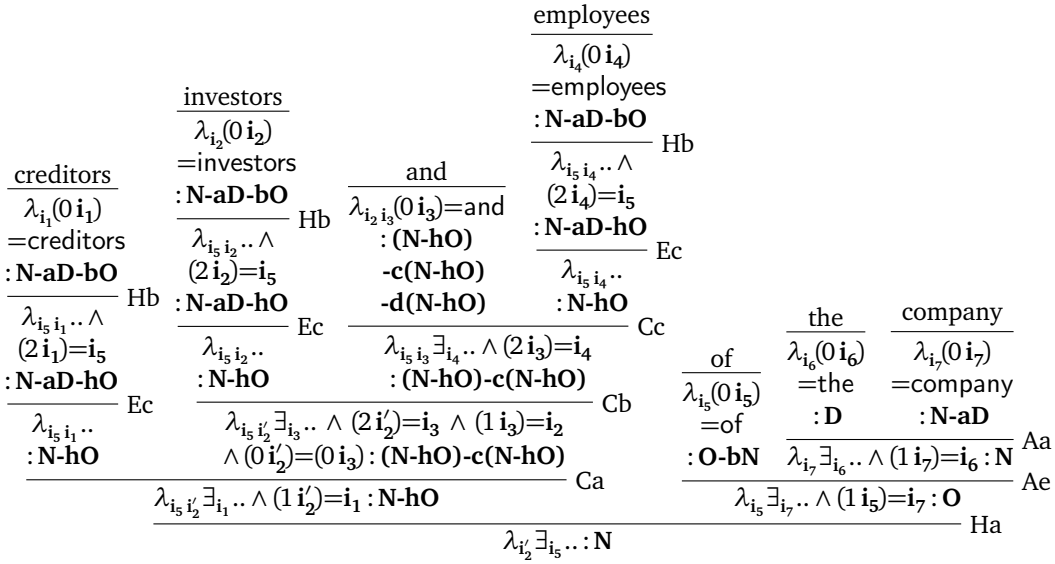


Figure 3: Example categorization of the noun phrase *creditors investors and employees of the company*. This derivation yields the following lexical relations: $(0\,\mathbf{i_1})$=creditors, $(0\,\mathbf{i_2})$=investors, $(0\,\mathbf{i'_2})=(0\,\mathbf{i_3})$=and, $(0\,\mathbf{i_4})$=employees, $(0\,\mathbf{i_5})$=of, $(0\,\mathbf{i_6})$=the, $(0\,\mathbf{i_7})$=company, and the following argument relations: $(2\,\mathbf{i_1})=\mathbf{i_5}$, $(2\,\mathbf{i_2})=\mathbf{i_5}$, $(2\,\mathbf{i_4})=\mathbf{i_5}$, $(1\,\mathbf{i'_2})=\mathbf{i_1}$, $(2\,\mathbf{i'_2})=\mathbf{i_3}$, $(1\,\mathbf{i_3})=\mathbf{i_2}$, $(2\,\mathbf{i_3})=\mathbf{i_4}$, $(2\,\mathbf{i_5})=\mathbf{i_7}$, $(1\,\mathbf{i_7})=\mathbf{i_6}$.

is associated with the variable of an existential quantifier. These existentially quantified variables can then be uniquely identified using numerical indices of words, and the numbered functions in lambda expressions $(v\,i)=j$ are interpreted as dependency relations assigning the v^{th} argument of i to be j.

This system has the attractive property that the same syntactic constraints can be assigned the same category in every context. This property is not shared by most categorial grammars: e.g. post-nominal and post-copular prepositional phrases often have different categories.

4 Mapping Treebank to a Generalized Categorial Grammar

The reannotation system described in this paper defines its target grammar in terms of a set of reannotation rules. These reannotation rules work within a script that traverses each bracketed sentence in a corpus by selecting each pair of matching brackets from the top of the tree to the bottom, then running a sed-like pattern substitution rule on each selection (see Figure 4). Such rules can implement local syntactic transformations, as well as certain non-local transformations like adding gap arguments to constituents containing a particular trace marker, for example. Reannotation of the categorial grammar evaluated in this paper requires about 150 such rules. These rules are modular and can be reused or modified to experiment with different syntactic analyses.

In order to make the trace and function labels in the Treebank accessible to a PCFG-based latent variable annotator, they must be incorporated into the syntactic categories of each tree and propagated from filler to gap constituents, creating a categorial grammar with local associations between parents and immediate children. First all trace annotations for interroga-

a) s/{(V-rN) <(WHNP)(-[0-9]+)([^ >]*)> (.*[-NONE- *T*\3].*)}/{\1 <N-rN\4> <V-gN\3 \5>}/;

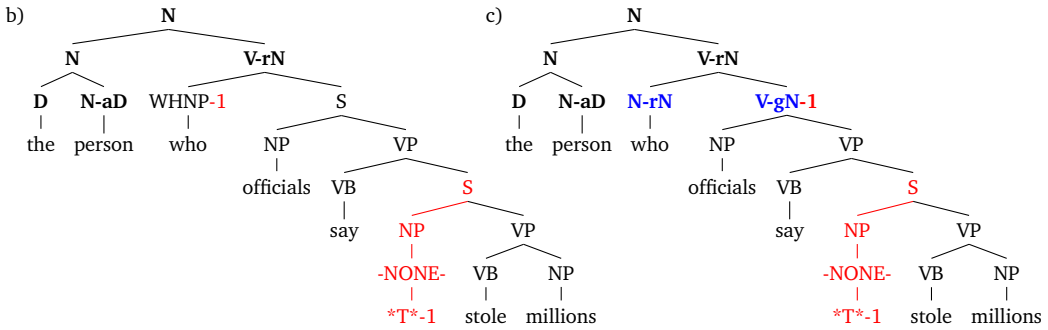


Figure 4: Sample sed-like reannotation rule introducing a gap tag at the top of a relative clause (a), and an application of this rule to the movement-based notation in the Penn Treebank (b) to produce a binary-branching categorial grammar derivation using gap arguments (c). Rules are applied to every constituent from the top of the tree down, using parentheses to delimit constituents above the current constituent, curly braces to delimit the current constituent, angle brackets to delimit child constituents, and square brackets to delimit constituents below children. Delimiters are then updated at every iteration.

tives, relative clauses, and topicalizations are transformed into gap arguments: **-gN** (for noun phrase gaps), **-g(R-aN)** (for adverbial phrase gaps), and **-gS** (for e.g. topicalized sentential gaps), which follow the category label for each constituent. Similar transformations localize it-clefts (*it seems that...*), tough constructions (*tough to cut*), parentheticals (*he/she said*), and certain types of inversion (*'it rained,' she said*), also using the gap operator **-g**. This is similar to the treatment of filler-gap constructions used in HPSG (Pollard and Sag, 1994).

Then, specifiers of head projections are annotated as initial arguments, e.g. **-aN** for nominal subjects, and complements of head projections are annotated as final arguments: e.g. **-bN** for transitive verbs, prepositions, and certain adjectives, **-bV** for sentential complements, **-bN-bN** for ditransitive verbs, etc. The **-h** operator is then used to propagate directional dependencies in right node raising (*they peeled and ate shrimp*). These are similar to the 'subcat' feature in HPSG, or to the left and right slash in categorial grammar accounts of specification, complementation and right node raising.

Then, interrogative and relative pronouns (e.g. *what* and *which*) are distinguished with **-iN** and **-rN** arguments, respectively, in order to regularize typical contexts for filler-gap traces.

Conjuncts are assigned **-c** and **-d** arguments to distinguish composition functions for conjuncts from composition functions for ordinary arguments. This distinction makes it possible for ordinary arguments to be shared among conjuncts.

The transform rules are defined as recursive rewrites that progress down the Treebank trees, propagating **-a**, **-b**, **-c**, **-d**, **-g**, **-h**, **-i**, and **-r** arguments as they go. Figure 4 shows a single step in a reannotation of the noun phrase *the person who officials say stole millions*. This set of recursive rules obtains about 94% coverage of the training set, consisting of sections 2–21 of the Penn Treebank. Trees not completely transformed by these rules are excluded from training. Since the system is evaluated on unbounded dependency annotations as described by Rimell et al. (2009), it is not necessary to apply this transformation to the test set.

In many cases the conversion to this categorial grammar replaces Treebank category labels with more general specifications. For example, in most contexts, adjective phrases, prepositional phrases, and progressive and passive verb phrases are replaced with a predicative category **A-aN**. This allows an HPSG-, GPSG- or CCG-like treatment of conjunction of any of these kinds of phrases with any other, following Sag et al. (1985), Gazdar et al. (1985), and Komagata (2002). This generalized predicative category is also used for noun phrase modifiers, and for noun phrases following the copular *be*. This annotation is compatible with additional specification of categories for adjective phrases (following words like *become*), or gerund phrases (following words like *start*), but these contexts cannot be reliably identified by the current reannotation.

In other cases, the conversion to categorial grammar distinguishes categories that are conflated in the Treebank. For example, the Treebank 'SBAR' category is distinguished into adjectival relative clauses **V-rN**, embedded questions **V-iN**, embedded inflected sentences **C**, embedded infinitival sentences **E**, nominal clauses **N**, adjectival modifier phrases **A-aN**, and adverbial modifier phrases **R-aN** (e.g. *because* . . .). Treebank 'S' categories are similarly distinguished into inflected sentences **V**, infinitival sentences **I**, base form sentences **B**, adjectival sentences (small clauses) **A**, and participial sentences **L**.[7]

Also, like other categorial grammars, this transform leaves trees in binary branching (Chomsky normal) form.

5 Evaluation

To evaluate the reannotation system described in this paper, a version of the Penn Treebank was annotated with a generalized categorial grammar as described in Section 3. This required about 150 reannotation rules, with about 20 rules for each of noun phrases, verb phrases, prepositional phrases, adjectival phrases, and adverbial phrases, and about 50 rules for various sentence types. This reannotation was applied to Sections 02 to 21 of the WSJ portion of the Penn Treebank (Marcus et al., 1993). The resulting corpus then underwent three iterations of latent variable annotation, using the split-merge algorithm of Petrov et al. (2006).[8] Each iteration of this algorithm splits the categories in the training corpus randomly in two, runs the inside-outside algorithm (expectation maximization) on the resulting trees to maximize the probability of the training set, then merges those categories that contributed the least to the maximization. The categorial grammar type-constructing operators were then used to extract semantic dependency relations for each parsed sentence.

This evaluation used development and test sets from Rimell et al. (2009) for tuning the annotation mappings and testing, respectively. The de Marneffe et al. (2006) dependencies used in these sets are deterministically mapped to the comparable numeric relations used by the current system.[9] Thus, the dependencies 'nsubj' and 'nsubjpass' are mapped to a '1' relation, and 'dobj', 'pobj', and 'obj2' are mapped to a '2' relation. The dependencies 'advmod', 'prep', and 'nn' are also mapped to a '1' relation with the direction of the dependency reversed.[10]

[7] It is important to note that these rules allow the word *that* to be treated as a relativizer rather than a relative pronoun, as it is in other categorial grammars, such as CCG.

[8] Under the current system, this number of split-merge iterations was empirically shown to yield the same results on development data as five iterations of split-merge, which is the recommended number for maximum accuracy without overfitting according to Petrov and Klein (2007).

[9] This simplification of dependency labels to numbers can be losslessly reversed by looking at the categories of the involved predicates.

[10] One anonymous reviewer points out that this reversal of direction for modifier dependencies is similar to that

	Obj RC	Obj Red	Sbj RC	Free	Obj Q	RNR	Sbj Embed	Total
Enju	47.3	65.9	82.1	**76.2**	32.5	47.1	32.9	54.4
C&C	**59.3**	62.6	80.0	72.6	27.5	**49.4**	22.4	53.6
Malt	40.7	50.5	**84.2**	70.2	16.2	39.7	23.5	46.4
MST	34.1	47.3	78.9	65.5	18.8	45.4	37.6	46.1
Stanford	22.0	1.1	74.7	64.3	41.2	45.4	10.6	38.1
DCU	23.1	41.8	56.8	46.4	27.5	40.8	5.9	35.7
Rasp	16.5	1.1	53.7	17.9	27.5	34.5	15.3	25.3
This system	52.7	**69.2**	68.4	69.0	**57.5**	26.4	**38.8**	**54.6**

Table 1: Unbounded dependency results compared to those of other systems studied by Rimell et al. (2009) and Nivre et al. (2010) over a variety of constructions: object extraction from relative clauses (Obj RC), object extraction from reduced relative clauses (Obj Red), subject extraction from relative clauses (Sbj RC), free relatives (Free), object wh-questions (Obj Q), right node raising (RNR), and subject extraction from embedded clauses (Sbj Embed). Evaluated parsers are C&C (Clark and Curran, 2007), Enju (Miyao and Tsujii, 2005), DCU (Cahill et al., 2004), Rasp (Briscoe et al., 2006), Stanford (Klein and Manning, 2003), MST (McDonald, 2006), Malt (Nivre et al., 2006a,b). This system used the Berkley parser (Petrov and Klein, 2007) run on the reannotated categorial grammar.

Due to differences between the de Marneffe et al. (2006) dependency representation and that of the current system, some deterministic modifications were required for evaluation against the Rimell et al. (2009) corpus.[11]

1. If the hypothesized target of a dependency is a conjunction, the dependencies to each of its conjuncts are hypothesized instead;

2. If the target of a dependency is a relativizer or a relative pronoun, the predicate it modifies is used in its place; and

3. If the source predicate of a dependency has a category of **O**, the predicate that depends on the hypothesized target is hypothesized as the target.

For example, in this categorial grammar analysis of the phrase *levels of health*, the word *of* heads a phrase of category **O**, with *health* as an argument, and *levels* depends directly on *health*, so the dependency that is evaluated is from *of* to *levels*. Finally, following Rimell et al. (2009), the current system counts both dependencies in a conjunction as having been captured if the first is correct and the conjunction is modelled correctly. If the source has the same type of relation to the target in the gold standard as was output by the current system, the dependency is counted as correct. The final results are tabulated in comparison to those in Rimell et al. (2009) and Nivre et al. (2010) in Table 1.[12]

described in dependency accounts of Tree Adjoining Grammars (Joshi, 1985; Candito and Kahane, 1998).

[11]This automated scoring makes the evaluation less generous than the manual output interpretations given in Rimell et al. (2009) and Nivre et al. (2010), but has the advantage of being easily reproducible.

[12]Note that one of the other systems was a variant of C&C which was given additional training on QuestionBank (Judge et al., 2006). This extra training and the results obtained from it were not used in the experimental description in this paper because i) other systems were not trained on this data and ii) extra training examples may distort the prior model to reduce probability of non-questions, which may have an adverse effect on overall parsing accuracy.

6 Conclusion

This paper has presented a system for automatically reannotating syntactically-annotated corpora for the purpose of refining linguistically-informed phrase structure analyses of various phenomena. The paper has also presented a generalized categorial grammar reannotation developed using this system which combines the transparent predicate-argument structure of a categorial grammar with the categorial generality of an HPSG-like system. The system achieves unbounded dependency parsing accuracy favorably comparable to all the systems recently studied by Rimell et al. (2009) and Nivre et al. (2010) on this same task.

This system and the generalized categorial grammar reannotation rules (and other scripts needed to produce the reannotated corpus) are available at:
http://sourceforge.net/projects/modelblocks/

Acknowledgements

Thanks to Laura Rimell for providing the unbounded dependency corpus and for her clarifications regarding the unbounded dependency recovery task. Thanks also to Bob Levine and Carl Pollard for insightful discussions related to this project, and to the anonymous reviewers for their comments. This work was funded by a Department of Linguistics Targeted Investment for Excellence (TIE) grant for collaborative interdisciplinary projects conducted during the academic year 2012-13.

References

Bach, E. (1981). Discontinuous constituents in generalized categorial grammars. *Proceedings of the Annual Meeting of the Northeast Linguistic Society (NELS)*, 11:1–12.

Bar-Hillel, Y. (1953). A quasi-arithmetical notation for syntactic description. *Language*, 29:47–58.

Blaheta, D. (2002). Handling noisy training and testing data. In *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing*.

Briscoe, T., Carroll, J., and Watson, R. (2006). The second release of the RASP system. In *Proceedings of the Interactive Demo Session of COLING/ACL-06*, Sydney, Australia.

Cahill, A., Burke, M., Genabith, J. V., and Way, A. (2004). Long-distance dependency resolution in automatically acquired wide-coverage PCFG-based lfg approximations. In *Proceedings of the 42nd Meeting of the ACL*, pages 320–327, Barcelona, Spain.

Campbell, R. (2004). Using linguistic principles to recover empty categories. In *In Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL-04)*, pages 645–652, Barcelona, Spain.

Candito, M.-H. and Kahane, S. (1998). Can the TAG derivation tree represent a semantic graph? In *Proceedings of the TAG+4 Workshop*, University of Pennsylvania.

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

Collins, M. (1997). Three generative, lexicalised models for statistical parsing. In *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics (ACL'97)*.

de Marneffe, M.-C., MacCartney, B., and Manning, C. D. (2006). Generating typed dependency parses from phrase structure parses. In *Proceedings of LREC 2006*.

Gazdar, G., Klein, E., Pullum, G., and Sag, I. (1985). *Generalized Phrase Structure Grammar*. Harvard University Press, Cambridge, MA.

Gildea, D. and Hockenmaier, J. (2003). Identifying semantic roles using combinatory categorial grammar. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Sapporo, Japan.

Hockenmaier, J. (2003). *Data and Models for Statistical Parsing with Combinatory Categorial Grammar*. PhD thesis, University of Edinburgh.

Joshi, A. K. (1985). How much context sensitivity is necessary for characterizing structural descriptions: Tree adjoining grammars. In D. Dowty, L. K. and Zwicky, A., editors, *Natural language parsing: Psychological, computational and theoretical perspectives*, pages 206–250. Cambridge University Press, Cambridge, U.K.

Judge, J., Cahill, A., and van Genabith, J. (2006). Questionbank: Creating a corpus of parse-annotated questions. In *Proceedings of the Joint Conference of the International Committee on Computational Linguistics and the Association for Computational Linguistics*.

Klein, D. and Manning, C. D. (2003). Accurate unlexicalized parsing. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 423–430, Sapporo, Japan.

Komagata, N. (2002). Coordination of unlike (?) categories: How not to distinguish categories.

Kubota, Y. and Levine, R. (2012). Gapping as like-category coordination. *Logical Aspects of Computational Linguistics*, (7351):Al135–150.

Lambek, J. (1958). The mathematics of sentence structure. *American mathematical monthly*, pages 154–170.

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

McDonald, R. (2006). *Discriminative Learning and Spanning Tree Algorithms for Dependency Parsing*. PhD thesis, University of Pennsylvania.

Mel'čuk, I. (1988). *Dependency syntax: theory and practice*. State University of NY Press, Albany.

Mihalicek, V. and Pollard, C. (2010). Distinguishing phenogrammar from tectogrammar simplifies the analysis of interrogatives. In *Formal Grammar*, pages 130–145.

Miyao, Y., Ninomiya, T., and Tsujii, J. (2004). Corpus-oriented grammar development for acquiring a head-driven phrase structure grammar from the Penn Treebank. In *Proceedings of the First International Joint Conference on Natural Language Processing (IJCNLP-04)*, pages 684—693, Hainan Island, China.

Miyao, Y. and Tsujii, J. (2005). Probabilistic disambiguation models for wide-coverage HPSG parsing. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 83–90, Michigan, Ann Arbor.

Nivre, J., Hall, J., and Nilsson, J. (2006a). Maltparser: A data-driven parser-generator for dependency parsing. In *In Proc. of LREC*, pages 2216–2219.

Nivre, J., Hall, J., Nilsson, J., Eryiğit, G., and Marinov, S. (2006b). Labeled pseudo-projective dependency parsing with support vector machines. In *Proceedings of the Tenth Conference on Computational Natural Language Learning*, CoNLL-X '06, pages 221–225, Stroudsburg, PA, USA. Association for Computational Linguistics.

Nivre, J., Rimell, L., McDonald, R., and Gómez-Rodríguez, C. (2010). Evaluation of dependency parsers on unbounded dependencies. In *Proceedings of the 23rd International Conference on Computational Linguistics (COLING'10)*, pages 833–841.

Oehrle, R. T. (1994). Term-labeled categorial type systems. *Linguistics and Philosophy*, 17(6):633–678.

Parsons, T. (1990). *Events in the Semantics of English*. MIT Press.

Petrov, S., Barrett, L., Thibaux, R., and Klein, D. (2006). Learning accurate, compact, and interpretable tree annotation. In *Proceedings of the 44th Annual Meeting of the Association for Computational Linguistics (COLING/ACL'06)*.

Petrov, S. and Klein, D. (2007). Improved inference for unlexicalized parsing. In *Proceedings of NAACL HLT 2007*, pages 404–411, Rochester, New York. Association for Computational Linguistics.

Pollard, C. and Sag, I. (1994). *Head-driven Phrase Structure Grammar*. University of Chicago Press, Chicago.

Rimell, L., Clark, S., and Steedman, M. (2009). Unbounded dependency recovery for parser evaluation. In *Proceedings of EMNLP 2009*, volume 2, pages 813–821.

Sag, I. A., Gazdar, G., Wasow, T., and Weisler, S. (1985). Coordination and how to distinguish categories. *Natural Language & Linguistic Theory*, 3:117–171.

Shieber, S. M., Schabes, Y., and Pereira, F. C. (1995). Principles and implementation of deductive parsing. *Journal of Logic Programming*, 24:3–36.

Steedman, M. (2000). *The syntactic process*. MIT Press/Bradford Books, Cambridge, MA.