Unlexicalised Hidden Variable Models of Split Dependency Grammars[*]

Gabriele Antonio Musillo
Department of Computer Science
and Department of Linguistics
University of Geneva
1211 Geneva 4, Switzerland
musillo4@etu.unige.ch

Paola Merlo
Department of Linguistics
University of Geneva
1211 Geneva 4, Switzerland
merlo@lettres.unige.ch

Abstract

This paper investigates transforms of split dependency grammars into unlexicalised context-free grammars annotated with hidden symbols. Our best unlexicalised grammar achieves an accuracy of 88% on the Penn Treebank data set, that represents a 50% reduction in error over previously published results on unlexicalised dependency parsing.

1 Introduction

Recent research in natural language parsing has extensively investigated probabilistic models of phrase-structure parse trees. As well as being the most commonly used probabilistic models of parse trees, probabilistic context-free grammars (PCFGs) are the best understood. As shown in (Klein and Manning, 2003), the ability of PCFG models to disambiguate phrases crucially depends on the expressiveness of the symbolic backbone they use.

Treebank-specific heuristics have commonly been used both to alleviate inadequate independence assumptions stipulated by naive PCFGs (Collins, 1999; Charniak, 2000). Such methods stand in sharp contrast to partially supervised techniques that have recently been proposed to induce hidden grammatical representations that are finer-grained than those that can be read off the parsed sentences in treebanks (Henderson, 2003; Matsuzaki et al., 2005; Prescher, 2005; Petrov et al., 2006).

[*]Part of this work was done when Gabriele Musillo was visiting the MIT Computer Science and Artificial Intelligence Laboratory, funded by a grant from the Swiss NSF (PBGE2-117146). Many thanks to Michael Collins and Xavier Carreras for their insightful comments on the work presented here.

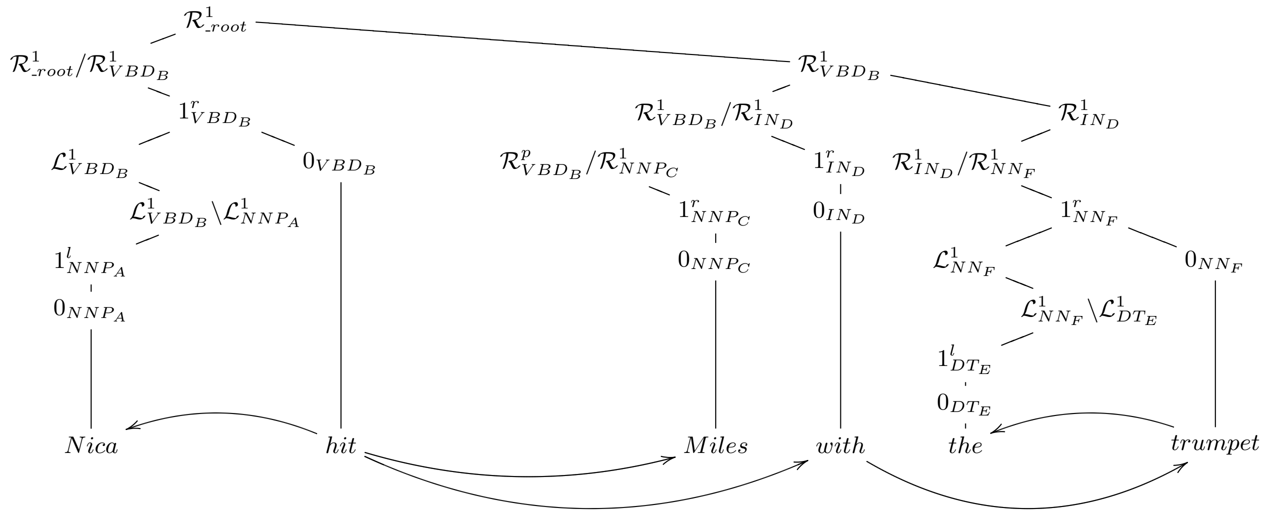
This paper presents extensions of such grammar induction techniques to dependency grammars. Our extensions rely on transformations of dependency grammars into efficiently parsable context-free grammars (CFG) annotated with hidden symbols. Because dependency grammars are reduced to CFGs, any learning algorithm developed for PCFGs can be applied to them. Specifically, we use the Inside-Outside algorithm defined in (Pereira and Schabes, 1992) to learn transformed dependency grammars annotated with hidden symbols. What distinguishes our work from most previous work on dependency parsing is that our models are not lexicalised. Our models are instead decorated with hidden symbols that are designed to capture both lexical and structural information relevant to accurate dependency parsing without having to rely on any explicit supervision.

2 Transforms of Dependency Grammars

Contrary to phrase-structure grammars that stipulate the existence of phrasal nodes, dependency grammars assume that syntactic structures are connected acyclic graphs consisting of vertices representing terminal tokens related by directed edges representing dependency relations. Such terminal symbols are most commonly assumed to be words. In our unlexicalised models reported below, they are instead assumed to be part-of-speech (PoS) tags. A typical dependency graph is illustrated in Figure 1 below.

Various projective dependency grammars exemplify the concept of split bilexical dependency grammar (SBG) defined in (Eisner, 2000). [1] SBGs are

[1]An SBG is a tuple $\langle V, W, L, R \rangle$ such that:



$\mathcal{R}^1_{\_root}$
$\mathcal{R}^1_{\_root}/\mathcal{R}^1_{VBD_B}$
$\mathcal{R}^1_{VBD_B}$
$1^r_{VBD_B}$
$\mathcal{R}^1_{VBD_B}/\mathcal{R}^1_{IN_D}$
$\mathcal{R}^1_{IN_D}$
$\mathcal{L}^1_{VBD_B}$
0_{VBD_B}
$\mathcal{R}^p_{VBD_B}/\mathcal{R}^1_{NNP_C}$
$1^r_{IN_D}$
$\mathcal{R}^1_{IN_D}/\mathcal{R}^1_{NN_F}$
$\mathcal{L}^1_{VBD_B}\backslash\mathcal{L}^1_{NNP_A}$
$1^r_{NNP_C}$
0_{IN_D}
$1^r_{NN_F}$
$1^l_{NNP_A}$
0_{NNP_C}
$\mathcal{L}^1_{NN_F}$
0_{NN_F}
0_{NNP_A}
$\mathcal{L}^1_{NN_F}\backslash\mathcal{L}^1_{DT_E}$
$1^l_{DT_E}$
0_{DT_E}
Nica
hit
Miles
with
the
trumpet

Figure 1: A projective dependency graph for the sentence *Nica hit Miles with the trumpet* paired with its second-order unlexicalised derivation tree annotated with hidden variables.

closely related to CFGs as they both define structures that are rooted ordered projective trees. Such a close relationship is clarified in this section.

It follows from the equivalence of finite automata and regular grammars that any SBG can be transformed into an equivalent CFG. Let $D = \langle V, W, L, R \rangle$ be a SBG and $G = \langle N, W, P, S \rangle$ a CFG. To transform D into G we to define the set P of productions, the set N of non-terminals, and the start symbol S as follows:

- For each v in W, transform the automaton L_v into a right-linear grammar G_{L_v} whose start symbol is $\mathcal{L}^1_v$; by construction, G_{L_v} consists of rules such as $\mathcal{L}^p_v \to u\, \mathcal{L}^q_v$ or $\mathcal{L}^p_v \to \epsilon$, where terminal symbols such as u belong to W and non-terminals such as $\mathcal{L}^p_v$ correspond to the states of the L_v automaton; include all ϵ-productions in P, and, if a rule such as $\mathcal{L}^p_v \to u\, \mathcal{L}^q_v$ is in G_{L_v}, include the rule $\mathcal{L}^p_v \to 2^l_u\, \mathcal{L}^q_v$ in P.
- For each v in V, transform the automaton R_v into a left-linear grammar G_{R_v} whose start symbol is $\mathcal{R}^1_v$; by construction, G_{R_v} consists of rules such as $\mathcal{R}^p_v \to \mathcal{R}^q_v\, u$ or $\mathcal{R}^p_v \to \epsilon$, where terminal symbols such as u belongs to W and non-terminals such as $\mathcal{R}^p_v$ correspond to the states of the R_v automaton; include all ϵ-productions in P, and, if a rule such as $\mathcal{R}^p_v \to \mathcal{R}^q_v\, u$ is in G_{R_v}, include the rule $\mathcal{R}^p_v \to \mathcal{R}^q_v\, 2^r_u$ in P.
- For each symbol 2^l_u occurring in P, include the productions $2^l_u \to \mathcal{L}^1_u\, 1^l_u$, $1^l_u \to 0_u\, \mathcal{R}^1_u$, and $0_u \to u$ in P; for each symbol 2^r_u in P, include the productions $2^r_u \to 1^r_u\, \mathcal{R}^1_u$, $1^r_u \to \mathcal{L}^1_u\, 0_u$, and $0_u \to u$ in P.
- Set the start symbol S to $\mathcal{R}^1_{\_root}$. [2]

Parsing CFGs resulting from such transforms runs in $O(n^4)$. The head index v decorating non-terminals such as 1^l_v, 1^r_v, 0_v, $\mathcal{L}^p_v$ and $\mathcal{R}^q_v$ can be computed in $O(1)$ given the left and right indices of the sub-string $w_{i,j}$ they cover. [3] Observe, however, that if 2^l_v or 2^r_v derives $w_{i,j}$, then v does not functionally depend on either i or j. Because it is possible for the head index v of 2^l_v or 2^r_v to vary from i to j, v has to be tracked by the parser, resulting in an overall $O(n^4)$ time complexity.

In the following, we show how to transform our $O(n^4)$ CFGs into $O(n^3)$ grammars by ap-

- V is a set of terminal symbols which include a distinguished element *_root*;
- L is a function that, for any $v \in W (= V - \{\_root\})$, returns a finite automaton that recognises the well-formed sequences in W^* of left dependents of v;
- R is a function that, for each $v \in V$, returns a finite automaton that recognises the well-formed sequences of right dependents in W^* for v.

[2]CFGs resulting from such transformations can further be normalised by removing the ϵ-productions from P.

[3]Indeed, if 1^l_v or 0_v derives $w_{i,j}$, then $v = i$; if 1^r_v derives $w_{i,j}$, then $v = j$; if $w_{i,j}$ is derived from $\mathcal{L}^p_v$, then $v = j + 1$; and if $w_{i,j}$ is derived from $\mathcal{R}^q_v$, then $v = i - 1$.

plying transformations, closely related to those in (McAllester, 1999) and (Johnson, 2007), that eliminate the 2^l_v and 2^r_v symbols.

We only detail the elimination of the symbols 2^r_v. The elimination of the 2^l_v symbols can be derived symmetrically. By construction, a 2^r_v symbol is the right successor of a non-terminal $\mathcal{R}^p_u$. Consequently, 2^r_v can only occur in a derivation such as

$$\alpha\,\mathcal{R}^p_u\,\beta \vdash \alpha\,\mathcal{R}^q_u\,2^r_v\,\beta \vdash \alpha\,\mathcal{R}^q_u\,1^r_v\,\mathcal{R}^1_v\,\beta.$$

To substitute for the problematic 2^r_v non-terminal in the above derivation, we derive the form $\mathcal{R}^q_u\,1^r_v\,\mathcal{R}^1_v$ from $\mathcal{R}^p_u/\mathcal{R}^1_v\;\;\mathcal{R}^1_v$ where $\mathcal{R}^p_u/\mathcal{R}^1_v$ is a new non-terminal whose right-hand side is $\mathcal{R}^q_u\,1^r_v$. We thus transform the above derivation into the derivation $\alpha\,\mathcal{R}^p_u\,\beta \vdash \alpha\,\mathcal{R}^p_u/\mathcal{R}^1_v\,\mathcal{R}^1_v\beta \vdash \alpha\,\mathcal{R}^q_u\,1^r_v\,\mathcal{R}^1_v\,\beta$. [4]

Because $u = i - 1$ and $v = j$ if $\mathcal{R}^p_u/\mathcal{R}^1_v$ derives $w_{i,j}$, and $u = j + 1$ and $v = i$ if $\mathcal{L}^p_u\backslash\mathcal{L}^1_v$ derives $w_{i,j}$, the parsing algorithm does not have to track any head indices and can consequently parse strings in $O(n^3)$ time.

The grammars described above can be further transformed to capture linear second-order dependencies involving three distinct head indices. A second-order dependency structure is illustrated in Figure 1 that involves two adjacent dependents, *Miles* and *with*, of a single head, *hit*.

To see how linear second-order dependencies can be captured, consider the following derivation of a sequence of right dependents of a head u:

$$\alpha\,\mathcal{R}^p_u/\mathcal{R}^1_v\,\beta \vdash \alpha\,\mathcal{R}^q_u\,1^r_v\,\beta \vdash \alpha\,\mathcal{R}^q_u/\mathcal{R}^1_w\,\mathcal{R}^1_w\;\;1^r_v\,\beta.$$

The form $\mathcal{R}^q_u/\mathcal{R}^1_w\;\mathcal{R}^1_w\;\;1_v$ mentions three heads: u is the the head that governs both v and w, and w precedes v. To encode the linear relationship between w and v, we redefine the right-hand side of $\mathcal{R}^p_u/\mathcal{R}^1_v$ as $\mathcal{R}^q_u/\mathcal{R}^1_w\;\langle\mathcal{R}^1_w,\;1^r_v\rangle$ and include the production $\langle\mathcal{R}^1_w,\;1^r_v\rangle \rightarrow\;\;\mathcal{R}^1_w\;1^r_v$ in the productions. The relationship between the dependents w and v of the head u is captured, because $\mathcal{R}^p_u/\mathcal{R}^1_v$ jointly generates $\mathcal{R}^1_w$ and 1^r_v. [5]

Any second-order grammar resulting from transforming the derivations of right and left dependents in the way described above can be parsed in $O(n^3)$, because the head indices decorating its symbols can be computed in $O(1)$.

In the following section, we show how to enrich both our first-order and second-order grammars with hidden variables.

3 Hidden Variable Models

Because they do not stipulate the existence of phrasal nodes, commonly used unlabelled dependency models are not sufficiently expressive to discriminate between distinct projections of a given head. Both our first-order and second-order grammars conflate distributionally distinct projections if they are projected from the same head. [6]

To capture various distinct projections of a head, we annotate each of the symbols that refers to it with a unique hidden variable. We thus constrain the distribution of the possible values of the hidden variables in a linguistically meaningful way. Figure 1 illustrates such constraints: the same hidden variable $_B$ decorates each occurrence of the PoS tag `VBD` of the head *hit*.

Enforcing such agreement constraints between hidden variables provides a principled way to capture not only phrasal information but also lexical information. Lexical pieces of information conveyed by a minimal projection such as 0_{VBD_B} in Figure 1 will consistently be propagated through the derivation tree and will condition the generation of the right and left dependents of *hit*.

In addition, states such as p and q that decorate non-terminal symbols such as $\mathcal{R}^p_u$ or $\mathcal{L}^q_u$ can also capture structural information, because they can encode the most recent steps in the derivation history. In the models reported in the next section, these states are assumed to be hidden and a distribution over their possible values is automatically induced.

4 Empirical Work and Discussion

The models reported below were trained, validated, and tested on the commonly used sections from the Penn Treebank. Projective dependency trees, ob-

[4]Symmetrically, the derivation $\alpha\,\mathcal{L}^p_u\,\beta \vdash \alpha\;\,2^l_v\,\mathcal{L}^q_u\,\beta \vdash \alpha\,\mathcal{L}^1_v\;1^l_v\;\mathcal{L}^q_u\;\beta$ involving the 2^l_v symbol is transformed into $\alpha\,\mathcal{L}^p_u\,\beta \vdash \alpha\,\mathcal{L}^1_v\;\;\mathcal{L}^p_u\backslash\mathcal{L}^1_v\,\beta \vdash \alpha\,\mathcal{L}^1_v\,1^l_v\,\mathcal{L}^q_u\,\beta$.

[5]Symmetrically, to transform the derivation of a sequence of left dependents of u, we redefine the right-hand side of $\mathcal{L}^p_u\backslash\mathcal{L}^1_v$ as $\langle 1^l_v, \mathcal{L}^1_w\rangle\;\mathcal{L}^q_u\backslash\mathcal{L}^1_w$ and include the production $\langle 1^l_v, \mathcal{L}^1_w\rangle \rightarrow 1^l_v\;\mathcal{L}^1_w$ in the set of rules.

[6]As observed in (Collins, 1999), an unambiguous verbal head such as *prove* bearing the `VB` tag may project a clause with an overt subject as well as a clause without an overt subject, but only the latter is a possible dependent of subject control verbs such as *try*.

Development Data – section 24	*per word*	*per sentence*
FOM: $q = 1, h = 1$	75.7	9.9
SOM: $q = 1, h = 1$	80.5	16.2
FOM: $q = 2, h = 2$	81.9	17.4
FOM: $q = 2, h = 4$	84.7	22.0
SOM: $q = 2, h = 2$	84.3	21.5
SOM: $q = 1, h = 4$	87.0	25.8

Test Data – section 23	*per word*	*per sentence*
(Eisner and Smith, 2005)	75.6	NA
SOM: $q = 1, h = 4$	88.0	30.6
(McDonald, 2006)	91.5	36.7

Table 1: Accuracy results on the development and test data set, where q denotes the number of hidden states and h the number of hidden values annotating a PoS tag involved in our first-order (FOM) and second-order (SOM) models.

tained using the rules stated in (Yamada and Matsumoto, 2003), were transformed into first-order and second-order structures. CFGs extracted from such structures were then annotated with hidden variables encoding the constraints described in the previous section and trained until convergence by means of the Inside-Outside algorithm defined in (Pereira and Schabes, 1992) and applied in (Matsuzaki et al., 2005). To efficiently decode our hidden variable models, we pruned the search space as in (Petrov et al., 2006). To evaluate the performance of our models, we report two of the standard measures: the *per word* and *per sentence* accuracy (McDonald, 2006).

Figures reported in the upper section of Table 1 measure the effect on accuracy of the transforms we designed. Our baseline first-order model ($q = 1, h = 1$) reaches a poor per word accuracy that suggests that information conveyed by bare PoS tags is not fine-grained enough to accurately predict dependencies. Results reported in the second line shows that modelling adjacency relations between dependents as second-order models do is relevant to accuracy. The third line indicates that annotating both the states and the PoS tags of a first-order model with two hidden values is sufficient to reach a performance comparable to the one achieved by a naive second-order model. However, comparing the results obtained by our best first-order models to the accuracy achieved by our best second-order model conclusively shows that first-order models exploit such dependencies to a much lesser extent. Overall, such results provide a first solution to the problem left open in (Johnson, 2007) as to whether second-order transforms are relevant to parsing accuracy or not.

The lower section of Table 1 reports the results achieved by our best model on the test data set and compare them both to those obtained by the only unlexicalised dependency model we know of (Eisner and Smith, 2005) and to those achieved by the state-of-the-art dependency parser in (McDonald, 2006). While clearly not state-of-the-art, the performance achieved by our best model suggests that massive lexicalisation of dependency models might not be necessary to achieve competitive performance. Future work will lie in investigating the issue of lexicalisation in the context of dependency parsing by weakly lexicalising our hidden variable models.

References

Eugene Charniak. 2000. A maximum-entropy-inspired parser. In *NAACL'00*.

Michael John Collins. 1999. *Head-Driven Statistical Models for Natural Language Parsing*. Ph.D. thesis, University of Pennsylvania.

Jason Eisner and Noah A. Smith. 2005. Parsing with soft and hard constraints on dependency length. In *IWPT'05*.

Jason Eisner. 2000. Bilexical grammars and their cubic-time parsing algorithms. In H.Bunt and A. Nijholt, eds., *Advances in Probabilistic and Other Parsing Technologies*, pages 29–62. Kluwer Academic Publishers.

Jamie Henderson. 2003. Inducing history representations for broad-coverage statistical parsing. In *NAACL-HLT'03*.

Mark Johnson. 2007. Transforming projective bilexical dependency grammars into efficiently-parsable cfgs with unfold-fold. In *ACL'06*.

Dan Klein and Christopher D. Manning. 2003. Accurate unlexicalized parsing. In *ACL'03*.

Takuya Matsuzaki, Yusuke Miyao, and Junichi Tsujii. 2005. Probabilistic CFG with latent annotations. In *ACL'05*.

David McAllester. 1999. A reformulation of eisner and satta's cubit time parser for split head automata grammars. http://ttic.uchicago.edu/d̃mcallester.

Ryan McDonald. 2006. *Discriminative Training and Spanning Tree Algorithms for Dependency Parsing*. Ph.D. thesis, University of Pennsylvania.

Fernando Pereira and Yves Schabes. 1992. Inside-outside reestimation form partially bracketed corpora. In *ACL'92*.

Slav Petrov, Leon Barrett Romain Thibaux, and Dan Klein. 2006. Learning accurate, compact, and interpretable tree annotation. In *ACL'06*.

Detlef Prescher. 2005. Head-driven PCFGs with latent-head statistics. In *IWPT'05*.

H. Yamada and Y. Matsumoto. 2003. Statistical dependency analysis with support vectore machines. In *IWPT'03*.