Wide-Coverage Semantic Representations from a CCG Parser

Johan Bos, Stephen Clark, Mark Steedman
School of Informatics, University of Edinburgh
{jbos,stevec,steedman}@inf.ed.ac.uk

James R. Curran
School of Information Technologies, University of Sydney
james@it.usyd.edu.au

Julia Hockenmaier
Institute for Research in Cognitive Science, University of Pennsylvania
juliahr@linc.cis.upenn.edu

Abstract

This paper shows how to construct semantic representations from the derivations produced by a wide-coverage CCG parser. Unlike the dependency structures returned by the parser itself, these can be used directly for semantic interpretation. We demonstrate that well-formed semantic representations can be produced for over 97% of the sentences in unseen WSJ text. We believe this is a major step towards wide-coverage semantic interpretation, one of the key objectives of the field of NLP.

1 Introduction

The levels of accuracy and robustness recently achieved by statistical parsers (e.g. Collins (1999), Charniak (2000)) have led to their use in a number of NLP applications, such as question-answering (Pasca and Harabagiu, 2001), machine translation (Charniak et al., 2003), sentence simplification (Carroll et al., 1999), and a linguist's search engine (Resnik and Elkiss, 2003). Such parsers typically return phrase-structure trees in the style of the Penn Treebank, but without traces and co-indexation. However, the usefulness of this output is limited, since the underlying meaning (as represented in a predicate-argument structure or logical form) is difficult to reconstruct from such skeletal parse trees.

In this paper we demonstrate how a wide-coverage statistical parser using Combinatory Categorial Grammar (CCG) can be used to generate semantic representations. There are a number of advantages to using CCG for this task. First, CCG provides "surface compositional" analysis of certain syntactic phenomena such as coordination and extraction, allowing the logical form to be obtained for such cases in a straightforward way. Second, CCG is a lexicalised grammar, and only uses a small number of semantically transparent combinatory rules to combine CCG categories. Hence providing a compositional semantics for CCG simply amounts to assigning semantic representations to the lexical entries and interpreting the combinatory rules. And third, there exist highly accurate, efficient and robust CCG parsers which can be used directly for this task (Clark and Curran, 2004b; Hockenmaier, 2003).

The existing CCG parsers deliver predicate argument structures, but not semantic representations that can be used for inference. The present paper seeks to extend one of these wide coverage parsers by using it to build logical forms suitable for use in various NLP applications that require semantic interpretation.

We show how to construct first-order representations from CCG derivations using the λ-calculus, and demonstrate that semantic representations can be produced for over 97% of the sentences in unseen WSJ text. The only other deep parser we are aware of to achieve such levels of robustness for the WSJ is Kaplan et al. (2004). The use of the λ-calculus is integral to our method. However, first-order representations are simply used as a proof-of-concept; we could have used DRSs (Kamp and Reyle, 1993) or some other representation more tailored to the application in hand.

There is some existing work with a similar motivation to ours. Briscoe and Carroll (2002) generate underspecified semantic representations from their robust parser. Toutanova et al. (2002) and Kaplan et al. (2004) combine statistical methods with a linguistically motivated grammar formalism (HPSG and LFG respectively) in an attempt to achieve levels of robustness and accuracy comparable to the Penn Treebank parsers (which Kaplan et al. do achieve). However, there is a key difference between these approaches and ours. In our approach the creation of the semantic representations forms a completely

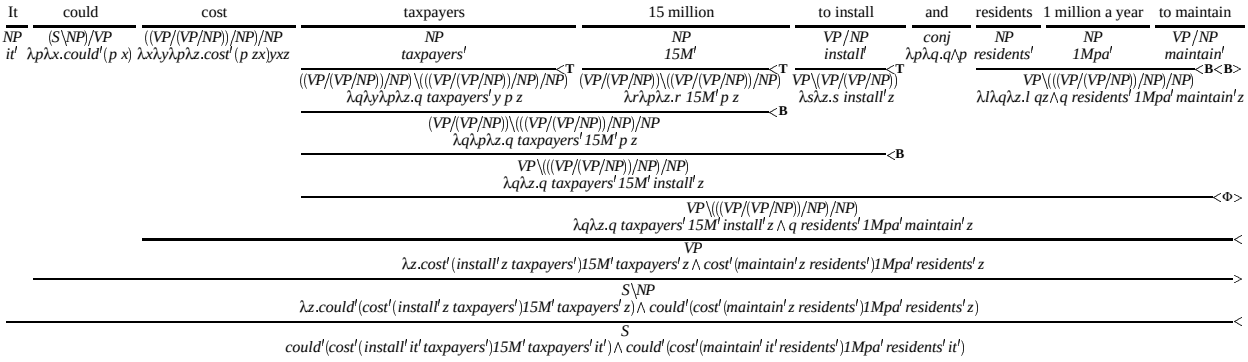

Figure 1: An example CCG derivation with a provisional semantics using predicate-argument structures

separate module to the syntax, whereas in the LFG and HPSG approaches the semantic representation forms an integral part of the grammar. This means that, in order for us to work with another semantic formalism, we simply have to modify the lexical entries with respect to the semantic component.

2 Combinatory Categorial Grammar

We assume familiarity with CCG (Steedman, 2000), an entirely type-driven lexicalized theory of grammar based on categorial grammar. CCG lexical entries pair a syntactic category (defining syntactic valency and directionality) with a semantic interpretation. For example, one of the categories for the verb *cost* can be written as follows, with a provisional Montague-style semantics expressed in terms of predicate-argument structure:[1]

$$((VP/(VP/NP))/NP)/NP \ : \\ \lambda x \lambda y \lambda p \lambda z.cost'(p\ zx)yxz$$

Combinatory rules project such lexical category-interpretation pairs onto derived category-interpretation pairs. The specific involvement in CCG of rules of functional composition (indexed $<\mathbf{B}$ and $>\mathbf{B}$ in derivations) and type-raising (indexed $<\mathbf{T}$ and $>\mathbf{T}$) allows very free derivation of non-standard constituents. This results in semantic interpretations that support the "surface compositional" analysis of relativization and coordination, as in Figure 1 for the sentence *It could cost taxpayers £15 million to install and BPC residents 1 million a year to maintain*.[2]

While the proliferation of surface constituents allowed by CCG adds to derivational ambiguity (since the constituent *taxpayers £15 million to install* is also allowed in the non-coordinate sentence *It could cost taxpayers £15 million to install*), previous work has shown that standard techniques from the statistical parsing literature can be used for practical wide-coverage parsing with state-of-the-art performance.

3 The Parser

A number of statistical parsers have recently been developed for CCG (Clark et al., 2002; Hockenmaier and Steedman, 2002b; Clark and Curran, 2004b). All of these parsers use a grammar derived from CCGbank (Hockenmaier and Steedman, 2002a; Hockenmaier, 2003), a treebank of normal-form CCG derivations derived semi-automatically from the Penn Treebank. In this paper we use the Clark and Curran (2004b) parser, which uses a log-linear model of normal-form derivations to select an analysis.

The parser takes a POS tagged sentence as input with a set of lexical categories assigned to each word. A CCG supertagger (Clark and Curran, 2004a) is used to assign the categories. The supertagger uses a log-linear model of the target word's context to decide which categories to assign. Clark and Curran (2004a) shows how dynamic use of the supertagger — starting off with a small number of categories assigned to each word and gradually increasing the number until an analysis is found — can lead to a highly efficient and robust parser.

The lexical category set used by the parser consists of those category types which occur at least 10 times in sections 2-21 of CCGbank, which results in a set of 409 categories. Clark and Curran (2004a) demonstrates that this relatively small set has high coverage on unseen data and can be used to create

[1] This semantic notation uses x, y, z and p, q, r, s as variables, identifies constants with primes and uses concatenation $a\ b$ to indicate application of a to b. Application is "left-associative," so abc is equivalent to $(ab)c$. The order of arguments in the predication is "wrapped", consistent with the facts of reflexive binding.

[2] Some details of the derivation and of the semantics of noun phrases are suppressed, since these are developed below. Some categories and interpretations are split across lines to save space.

a robust and accurate parser. The relevance of a relatively small category set is that, in order to obtain semantic representations for a particular formalism, only 409 categories have to be annotated.

The parser uses the CKY chart-parsing algorithm from Steedman (2000). The combinatory rules used by the parser are functional application (forward and backward), generalised forward composition, backward composition, generalised backward-crossed composition, and type raising. There is also a coordination rule which conjoins categories of the same type.

The parser also uses a number of unary type-changing rules (Hockenmaier and Steedman, 2002a) and punctuation rules taken from CCGbank. An example of a type-changing rule used by the parser is the following, which takes a passive form of a verb and creates a nominal modifier:

$$S[pss]\backslash NP \Rightarrow NP\backslash NP \quad (1)$$

This rule is used to create *NP*s such as *the role played by Kim Cattrall*. An example of a comma rule is the following:

$$S/S \ , \Rightarrow S/S \quad (2)$$

This rule takes a sentential modifier followed by a comma and returns a sentential modifier of the same type.

Type-raising is applied to the categories *NP*, *PP* and $S[adj]\backslash NP$ (adjectival phrase), and is implemented by adding the relevant set of type-raised categories to the chart whenever an *NP*, *PP* or $S[adj]\backslash NP$ is present. The sets of type-raised categories are based on the most commonly used type-raising rule instantiations in sections 2-21 of CCGbank, and currently contain 8 type-raised categories for *NP* and 1 each for *PP* and $S[adj]\backslash NP$.

For a given sentence, the automatically extracted grammar can produce a very large number of derivations. Clark and Curran (2003) and Clark and Curran (2004b) describe how a packed chart can be used to efficiently represent the derivation space, and also efficient algorithms for finding the most probable derivation. The parser uses a log-linear model over normal-form derivations.[3] Features are defined in terms of the local trees in the derivation, including lexical head information and word-word dependencies. The normal-form derivations in CCGbank provide the gold standard training data.

For a given sentence, the output of the parser is a set of syntactic dependencies corresponding to the most probable derivation. However, for this paper the parser has been modified to simply output the derivation in the form shown in Figure 2, which is the input for the semantic component.

[3] A *normal-form* derivation is one which only uses type-raising and function composition when necessary.

4 Building Semantic Representations

4.1 Semantic Formalism

Our method for constructing semantic representations can be used with many different semantic formalisms. In this paper we use formulas of first-order logic with a neo-Davidsonian analysis of events. We do not attempt to cover all semantic phenomena; for example, we do not currently deal with the resolution of pronouns and ellipsis; we do not give a proper analysis of tense and aspect; we do not distinguish between distributive and collective readings of plural noun phrases; and we do not handle quantifier scope ambiguities.

The following first-order formula for the sentence *A spokesman had no comment* demonstrates the representation we use:

$$\exists x(\text{spokesman}(x) \wedge \forall y(\text{comment}(y) \rightarrow \neg\exists e(\text{have}(e) \wedge \text{agent}(e,x) \wedge \text{patient}(e,y)))).$$

The tool that we use to build semantic representations is based on the lambda calculus. It can be used to mark missing semantic information from natural language expressions in a principled way using λ, an operator that binds variables ranging over various semantic types. For instance, a noun phrase like *a spokesman* can be given the λ-expression

$$\lambda p.\exists x(\text{spokesman}(x) \wedge (p@x))$$

where the @ denotes functional application, and the variable p marks the missing information provided by the verb phrase. This expression can be combined with the λ-expression for *lied*, using functional application, yielding the following expression:

$$\lambda p.\exists x(\text{spokesman}(x) \wedge (p@x))@ \lambda y.\exists e(\text{lie}(e) \wedge \text{agent}(e,y)).$$

β-conversion is the process of eliminating all occurrences of functional application by substituting the argument for the λ-bound variables in the functor. β-conversion turns the previous expression into a first-order translation for *A spokesman lied*:

$$\exists x(\text{spokesman}(x) \wedge \exists e(\text{lie}(e) \wedge \text{agent}(e,x))).$$

The resulting semantic formalism is very similar to the type-theoretic language L_λ (Dowty et

al., 1981). However, we merely use the lambda-calculus as a tool for constructing semantic representations, rather as a formal tool for model-theoretic interpretation. As already mentioned, we can use the same method to obtain, for example, Discourse Representation Structures (Kuschert, 1999), or underspecified semantic representations (Bos, 2004) to deal with quantifier scope ambiguities.

4.2 Method and Algorithm

The output of the parser is a tree representing a CCG derivation, where the leaves are lexical items and the nodes correspond to one of the CCG combinatory rules, a unary type-changing rule, a type-raising rule, or one of the additional miscellaneous rules discussed earlier. Mapping the CCG derivation into a semantic representation consists of the following tasks:

1. assigning semantic representations to the lexical items;
2. reformulating the combinatory rules in terms of functional application;
3. dealing with type-raising and type-changing rules;
4. applying β-conversion to the resulting tree structure.

Lexical items are ordered pairs consisting of the CCG category and a lemmatised wordform. This information is used to assign a λ-expression to the leaf nodes in the tree. For most open-class lexical items we use the lemma to instantiate the lexical semantics, as illustrated by the following two examples (intransitive verbs and adjectives):

$\langle$`S[dcl]\NP`, *walk*$\rangle$ =
λqλu.q@λx.∃e(walk(e)∧agent(e,x)∧u@e)

$\langle$`N/N`, *big*$\rangle$ =
λpλx.(big(x)∧p@x)

For closed-class lexical items, the lexical semantics is spelled out for each lemma individually, as in the following two examples:

$\langle$`(S[X]\NP)\(S[X]\NP)`, *not*$\rangle$ =
λvλqλf.¬((v@q)@f)

$\langle$`NP[nb]/N`, *all*$\rangle$ =
λpλq.∀x(p@x→q@x)

The second task deals with the combinatory rules. The rules we currently use are forward and backward application (FAPP, BAPP), generalised forward composition (FCOMP), backward composition (BCOMP), and generalised backward-crossed composition (BCROSS).

$\text{FAPP}(x,y) = (x@y)$
$\text{BAPP}(x,y) = (y@x)$
$\text{FCOMP}(x,y) = \lambda\overline{u}.(x@(\overline{u}@y))$
$\text{BCOMP}(x,y) = \lambda u.(y@(u@x))$
$\text{BCROSS}(x,y) = \lambda\overline{u}.(y@(x@\overline{u}))$

The type-raising and type-changing rules are dealt with by looking up the specific rule and replacing it with the resulting semantics. For instance, the rule that raises category `NP` to `S[X]/(S[X]\NP)` converts the semantics as follows:

TYPERAISE(`NP`, `S[X]/(S[X]\NP)`, x)
= λvλe.((v@x)@e)

The following type-changing rule applies to the lexical semantics of categories of type `N` and converts them to `NP`:

TYPECHANGE(`N`, `NP`, y)
= λp.∃x(y@x∧p@x)

Tasks 1–3 are implemented using a recursive algorithm that traverses the derivation and returns a λ-expression. Note that the punctuation rules used by the parser do not contribute to the compositional semantics and are therefore ignored.

Task 4 reduces the λ-expression to the target representation by applying β-conversion. In order to maintain correctness of this operation, the functor undergoes α-conversion (renaming all bound variables for new occurrences) before substitution takes place. β-conversion is implemented using the tools provided by Blackburn and Bos (2003).

4.3 Results

There are a number of possible ways to evaluate the semantic representations output by our system. The first is to calculate the coverage — that is, the percentage of syntactic parses which can be given some analysis by the semantic component. The second is to evaluate the accuracy of the semantic representations; the problem is that there is not yet an accepted evaluation metric which can be applied to such representations.

There is, however, an accepted way of evaluating the syntactic component of the system, namely to calculate precision and recall figures for labelled syntactic dependencies (Clark et al., 2002). Given

```
bapp('S[dcl]',
     bapp('NP',
          fapp('NP[nb]',
               leaf('NP[nb]/N','the'),
               fapp('N',
                    leaf('N/N',school-board'),
                    leaf('N','hearing'))),
          fapp('NP\NP',
               bapp('(NP\NP)/S[dcl]',
                    leaf('(NP\NP)/NP','at'),
                    leaf('((NP\NP)/S[dcl])\((NP\NP)/NP)','which')),
               bapp('S[dcl]',
                    leaf('NP','she'),
                    fapp('S[dcl]\NP',
                         leaf('(S[dcl]\NP)/(S[pss]\NP)','was'),
                         leaf('S[pss]\NP','dismissed')))),
     fapp('S[dcl]\NP',
          leaf('(S[dcl]\NP)/(S[pss]\NP)','was'),
          fapp('S[pss]\NP',
               leaf('(S[pss]\NP)/PP','crowded'),
               fapp('PP',
                    leaf('PP/NP','with'),
                    bapp('NP',
                         lex('NP',leaf('N','students')),
                             conj('conj','NP','NP\NP',
                                  leaf('conj','and'),
                                  lex('NP',leaf('N','teachers')))))))).

some A ((school-board[A] & hearing[A]) & some B (female[B] & some C
(dismiss[C] & (patient[C,B] & (at[A,C] & some D (crowd[D] & (patient[D,A]
& ((some E (student[E] & with[D,E]) & some F (teacher[F] & with[D,F])) &
event[D])))))))))
```

Figure 2: Parser output and semantic representation for the example sentence:
The school-board hearing at which she was dismissed was crowded with students and teachers

that the CCG parser produces dependencies which are essentially predicate-argument dependencies, the accuracy of the syntactic component should be a good indication of the accuracy of the semantics, especially given the transparent interface between syntax and semantics used by our system. Hence we report coverage figures in this paper, and repeat figures for dependency recovery from an earlier paper.

We do not evaluate the accuracy of the system output directly, but we do have a way of checking the well-formedness of the semantic representations. (The well-formedness of the representation does not of course guarantee the correctness of the output.) If the semantic representation fails to β-convert, we know that there are type conflicts resulting from either: incorrect semantics assigned to some lexical entries; incorrect interpretation of one of the combinatory rules; or an inconsistency in the output of the syntactic component.

We assigned lexical semantics to the 245 most frequent categories from the complete set of 409, and implemented 4 of the type-raising rules, and the 10 unary type-changing rules, used by the parser. We used section 00 from CCGbank for development purposes; section 23 (2,401 sentences) was used as the test set. The parser provides a syntactic analysis for 98.6% of the sentences in section 23. The accuracy of the parser is reported in Clark and Curran (2004b): 84.6% F-score over labelled dependencies for section 23. Of the sentences the parser analyses, 92.3% were assigned a semantic representation, all of which were well-formed. The output of the system for an example sentence is given in Figure 2.

The reason for the lack of complete coverage is that we did not assign semantic representations to the complete set of lexical categories. In future work we will cover the complete set, but as a simple remedy we have implemented the following robustness strategy: we assign a semantic template to parts of the tree that could not be analysed. For example, the template for the *NP* category is $\lambda p.\exists x(p@x)$.

This was done for the 10 most frequent categories and results in a coverage of 98.6%.

Although we expect the accuracy of the semantic representations to mirror those of the syntactic component, and therefore be useful in NLP applications, there is still a small number of errors arising from different sources. First, some constructions are incorrectly analysed in CCGbank; for example, appositives in CCGbank are represented as coordinate constructions (Hockenmaier, 2003). Second, errors are introduced by the semantic construction component; for example, the non-head nouns in a noun-noun compound are currently treated as modifiers of the head noun, in the same way as adjectives. And finally, the parser introduces errors because of incomplete coverage of the lexicon, and mistakes due to the parsing model. We expect general improvements in statistical parsing technology will further improve the accuracy of the parser, and we will further develop the semantic component.

5 Conclusions and Future Work

This paper has demonstrated that we can construct semantic representations using a wide-coverage CCG parser, with a coverage of over 97% on unseen WSJ sentences. We believe this is a major step towards wide-coverage semantic interpretation, one of the key objectives of the field of NLP.

The advantages of our approach derive largely from the use of CCG. The lexicalised nature of the formalism means that our system has a high degree of modularity, with separate syntactic and semantic components.

We have shown how to construct simple first-order semantic representations from CCG derivations. We have not dealt with all semantic phenomena, such as quantifier scope ambiguities and anaphora resolution. In future work we will investigate using underspecified semantic representations. The utility of our system for NLP applications will be tested by integration with an existing open-domain Question-Answering system (Leidner et al., 2003).

We will also investigate the construction of a treebank of semantic representations derived automatically from CCGbank. Previous work, such as Liakata and Pulman (2002) and Cahill et al. (2003), has attempted to generate semantic representations from the Penn Treebank. Cahill et al. use a translation of the Treebank to LFG F-structures and quasi-logical forms. An advantage of our approach is that our system for constructing semantic representations, whatever semantic formalism is used, can be applied directly to the derivations in CCGbank.

Acknowledgements

This research was partially supported by EPSRC grant GR/M96889.

References

Patrick Blackburn and Johan Bos. 2003. Representation and Inference for Natural Language. A First Course in Computational Semantics. Draft available at `http://www.comsem.org`, June.

Johan Bos. 2004. Computational semantics in discourse: Underspecification, resolution, and inference. *Journal of Logic, Language and Information*, 12(2).

Ted Briscoe and John Carroll. 2002. Robust accurate statistical annotation of general text. In *Proceedings of the 3rd LREC Conference*, pages 1499–1504, Las Palmas, Gran Canaria.

Aoife Cahill, Mairead McCarthy, Josef van Genabith, and Andy Way. 2003. Quasi-Logical Forms from F-Structures for the Penn Treebank. In Harry Bunt, Ielka van der Sluis, and Roser Morante, editors, *Proceedings of the Fifth International Workshop on Computational Semantics (IWCS-5)*, pages 55–71. Tilburg University.

J. Carroll, G. Minnen, D. Pearce, Y. Canning, S. Devlin, and J. Tait. 1999. Simplifying text for language-impaired readers. In *Proceedings of the 9th Meeting of EACL*, pages 269–270, Bergen, Norway.

Eugene Charniak, Kevin Knight, and Kenji Yamada. 2003. Syntax-based language models for machine translation. In *Proceedings of the MT Summit IX*, New Orleans, Louisiana.

Eugene Charniak. 2000. A maximum-entropy-inspired parser. In *Proceedings of the 1st Meeting of the NAACL*, pages 132–139, Seattle, WA.

Stephen Clark and James R. Curran. 2003. Log-linear models for wide-coverage CCG parsing. In *Proceedings of the EMNLP Conference*, pages 97–104, Sapporo, Japan.

Stephen Clark and James R. Curran. 2004a. The importance of supertagging for wide-coverage CCG parsing. In *Proceedings of the 20th International Conference on Computational Linguistics (COLING-04) (to appear)*, Geneva, Switzerland.

Stephen Clark and James R. Curran. 2004b. Parsing the WSJ using CCG and log-linear models. In *Proceedings of the 42nd Meeting of the ACL (to appear)*, Barcelona, Spain.

Stephen Clark, Julia Hockenmaier, and Mark Steedman. 2002. Building deep dependency structures with a wide-coverage CCG parser. In *Proceedings of the 40th Meeting of the ACL*, pages 327–334, Philadelphia, PA.

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

David R. Dowty, Robert E. Wall, and Stanley Peters. 1981. *Introduction to Montague Semantics*. Studies in Linguistics and Philosophy. D. Reidel Publishing Company.

Julia Hockenmaier and Mark Steedman. 2002a. Acquiring compact lexicalized grammars from a cleaner treebank. In *Proceedings of the Third LREC Conference*, pages 1974–1981, Las Palmas, Spain.

Julia Hockenmaier and Mark Steedman. 2002b. Generative models for statistical parsing with Combinatory Categorial Grammar. In *Proceedings of the 40th Meeting of the ACL*, pages 335–342, Philadelphia, PA.

Julia Hockenmaier. 2003. *Data and Models for Statistical Parsing with Combinatory Categorial Grammar*. Ph.D. thesis, University of Edinburgh.

Hans Kamp and Uwe Reyle. 1993. *From Discourse to Logic; An Introduction to Modeltheoretic Semantics of Natural Language, Formal Logic and DRT*. Kluwer, Dordrecht.

Ronald M. Kaplan, Stefan Riezler, Tracy H. King, John T. Maxwell III, Alexander Vasserman, and Richard Crouch. 2004. Speed and accuracy in shallow and deep stochastic parsing. In *Proceedings of the HLT/NAACL Conference*, Boston, MA.

Susanna Kuschert. 1999. *Dynamic Meaning and Accommodation*. Ph.D. thesis, Universität des Saarlandes.

Jochen L. Leidner, Johan Bos, Tiphaine Dalmas, James R. Curran, Stephen Clark, Colin J. Bannard, Mark Steedman, and Bonnie Webber. 2003. The QED open-domain answer retrieval system for TREC 2003. In *Proceedings of the Twelfth Text Retrieval Conference (TREC 2003)*, pages 595–599, Gaithersburg, MD.

Maria Liakata and Stephen Pulman. 2002. From trees to predicate-argument structures. In Shu-Chuan Tseng, editor, *COLING 2002. Proceedings of the 19th International Conference on Computational Linguistics*, pages 563–569. Taipei, Taiwan.

Marius Pasca and Sanda Harabagiu. 2001. High performance question/answering. In *Proceedings of the ACL SIGIR Conference on Research and Development in Information Retrieval*, pages 366–374, New Orleans LA.

Philip Resnik and Aaron Elkiss. 2003. The linguist's search engine: Getting started guide. Technical Report LAMP-TR-108/CS-TR-4541/UMIACS-TR-2003-109, University of Maryland, College Park, MA.

Mark Steedman. 2000. *The Syntactic Process*. The MIT Press, Cambridge, MA.

Kristina Toutanova, Christopher Manning, Stuart Shieber, Dan Flickinger, and Stephan Oepen. 2002. Parse disambiguation for a rich HPSG grammar. In *Proceedings of the First Workshop on Treebanks and Linguistic Theories*, pages 253–263, Sozopol, Bulgaria.