Lexicalized Grammar Acquisition

Yusuke Miyao† **Takashi Ninomiya**†‡ **Jun'ichi Tsujii**†‡
†Department of Computer Science, University of Tokyo
Hongo 7-3-1, Bunkyo-ku, Tokyo 113-0033 JAPAN
‡CREST, JST (Japan Science and Technology Corporation)
Honcho 4-1-8, Kawaguchi-shi, Saitama 332-0012 JAPAN
{yusuke,ninomi,tsujii}@is.s.u-tokyo.ac.jp

Abstract

This paper presents a formalization of automatic grammar acquisition that is based on lexicalized grammar formalisms (e.g. LTAG and HPSG). We state the conditions for the consistent acquisition of a unique lexicalized grammar from an annotated corpus.

1 Introduction

Linguistically motivated and computationally oriented grammar theories take the form of *lexicalized grammar formalisms*; examples include Lexicalized Tree Adjoining Grammar (LTAG) (Schabes et al., 1988), Combinatory Categorial Grammar (Steedman, 2000), and Head-driven Phrase Structure Grammar (HPSG) (Sag and Wasow, 1999). They have been a great success in terms of linguistic analysis and efficiency in the parsing of real-world texts. However, such grammars have not generally been considered suitable for the syntactic analysis within practical NLP systems because considerable effort is required to develop and maintain lexicalized grammars that are both robust and provide broad coverage.

One novel approach to grammar development is based on the automatic acquisition of lexicalized grammars from annotated corpora. Since lexicalized grammars represent grammatical constraints with a few grammar rules and a large number of lexical entries, the rules are quite easy to write but the construction of a lexicon is unrealistic. The idea in this study is to automatically obtain the lexical entries from an annotated corpus, which will greatly reduce the cost of building the grammar.

The approach has the following advantages. First, the grammars obtained will be robust because appropriate lexical entries are consistently acquired even for constructions beyond grammar developers' intuition. Secondly, the grammar developers are simply required to annotate the closed set of the training corpus, where various heuristic and statistical methods are applicable. Consistency between the grammar rules and the obtained lexical entries is assured independently of the methods of annotation. Lastly, the validity of the grammar theories is evaluated on real-world texts. A degree of low coverage by a linguistically motivated grammar does not necessarily reflect inadequacy of the grammar theories; a lack of appropriate lexical entries may also be responsible. The analysis of obtained grammars gives us grounds for discussing the pros and cons of the theories.

The studies on the extraction of LTAG (Xia, 1999; Chen and Vijay-Shanker, 2000; Chiang, 2000) and CCG (Hockenmaier and Steedman, 2002) represent the first attempts at the acquisition of linguistically motivated grammars from annotated corpora. Those studies are limited to specific formalisms, and can be interpreted as instances of our approach as described in Section 3. This paper does not describe any concrete algorithms for grammar acquisition that depend on specific grammar formalisms. The contribution of our work is to formally state the conditions required for the acquisition of lexicalized grammars and to demonstrate that it can be applied to lexicalized grammars other than LTAG and CCG, such as HPSG.

2 Lexicalized Grammars

In this section, we define the general form of lexicalized grammars including LTAG (Schabes et al., 1988) and HPSG (Sag and Wasow, 1999). The concepts behind a lexicalized grammar are that i) *grammar rules* represent general grammatical constructions in the language while ii) *lexical entries* describe word-specific lexical/syntactic constraints. Let W be a set of all words and C a set of linguistic representations (e.g. the tree structures of LTAG or typed feature structures of HPSG). We can formally define a lexicalized grammar in the following way.

Definition 1 (Lexicalized grammar) *A lexicalized grammar is a tuple $G = \langle L, R \rangle$, where L is a lexicon, i.e. $L \subset W \times C$, and R is a set of grammar rules, i.e. $r \in R$ is a partial function: $C \times C \rightarrow C$.*

In what follows, we assume that all grammar rules are binary for simplicity.

Parsing with a lexicalized grammar is the process of applying grammar rules to lexical entries. Since we assume that the rules are binary, the history of the process constitutes a binary-branching tree; we define such structures in this paper as *constituent structures*[1] (Miller, 2000).

Definition 2 (Constituent structure) *Given a sentence* $\mathbf{w}$, *a constituent structure of* $\mathbf{w}$ *is the least set* $\Gamma \subset W^*$ *satisfying i)* $\mathbf{w} \in \Gamma$ *and ii)* $\forall\gamma \in \Gamma.(|\gamma| > 1 \rightarrow \exists!\gamma_1, \gamma_2 \in \Gamma.(\gamma = \gamma_1\gamma_2))$

The first condition represents inclusion of the top node in the structure, and the second constrains the terminals (words) to a linear order.

The process of parsing is then depicted by a constituent structure labeled with grammar rules, which we call *a derivation history*.

Definition 3 (Derivation history) *Given a sentence* $\mathbf{w}$, a derivation history *is a tuple* $\Upsilon = \langle \Gamma, \rho \rangle$, *where* Γ *is a constituent structure of* $\mathbf{w}$ *and* ρ *is a function:* $\Gamma \rightarrow R$. *Derivation history* $\Upsilon = \langle \Gamma, \rho \rangle$ *is* a well-formed derivation history *iff there exists a function* ξ *satisfying the following conditions.*

- $\forall w \in \mathbf{w}. \exists \langle w, c \rangle \in L \wedge \xi(\langle w \rangle) = c$
- $\forall \gamma \in \Gamma. \exists \gamma_1, \gamma_2 \in \Gamma. \gamma = \gamma_1\gamma_2 \wedge \xi(\gamma) = \rho(\gamma)(\xi(\gamma_1), \xi(\gamma_2))$

Let α_i be $\xi(\gamma_i)$ or γ_i, we denote $\xi(\gamma) \Rightarrow \alpha_1 \ldots \alpha_n$ when $\gamma = \gamma_1 \ldots \gamma_n$.

The results of parsing (e.g. derived trees of LTAG and phrasal signs of HPSG) by a lexicalized grammar are outputs of the application of rules according to derivation histories.

Definition 4 (Parse result) *Given a sentence* $\mathbf{w}$, $c_s \in C$ *is* a parse result *of* $\mathbf{w}$ *iff* $c_s \Rightarrow \mathbf{w}$ *for some well-formed derivation history.*

Given the above definitions, our task is to obtain lexical entries $\langle w, c \rangle \in L$ from a parsed corpus, i.e., parse results $c_s \in C$. The idea is to make derivation histories from a parsed corpus, and reduce the parse results into lexical entries by traversing the derivation histories.

3 Grammar Acquisition

Given Definitions 1, 3 and 4, we can determine unique lexical entries from a parse result if a derivation history is given and the grammar rules $r \in R$ are injective functions, i.e., $\forall c. \exists c_1, c_2, c'_1, c'_2. (c = r(c_1, c_2) \wedge c = r(c'_1, c'_2)) \rightarrow (c_1 = c'_1 \wedge c_2 = c'_2)$

Theorem 1 (Grammar acquisition) *Given* $c_s \in C$ *as the parse result of sentence* $\mathbf{w}$, *and* Υ *as its derivation history, lexical entries for* $\mathbf{w}$ *are uniquely determined if all grammar rules* $r \in R$ *are injective functions. That is,* $\exists! c_1, \ldots, c_n \in C. (\forall i \leq n. \langle w_i, c_i \rangle \in L \wedge c_s \Rightarrow c_1 \ldots c_n)$.

Proof. *We construct lexical entries by inversely applying the grammar rules to the parse result. Since the grammar rules are injective, we are able to uniquely determine the inputs for any rule application. That is,* $\forall \gamma \in \Gamma. \exists! c \in C. \exists \beta_1\beta_2 \in C^*. c_s \Rightarrow \beta_1 c \beta_2 \wedge c \Rightarrow \gamma$. *We prove this lemma by the mathematical induction of the length of* γ.

1. *When* $|\gamma| = |\mathbf{w}|$, $\gamma = \mathbf{w}$ *according to Definition 2. This case is trivial.*
2. *When* $|\gamma| = k$, *we assume that the lemma is true for* $\forall i > k$. *From Definition 2,* $\exists! \gamma', \gamma'' \in \Gamma. \gamma' = \gamma\gamma'' \vee \gamma' = \gamma''\gamma$.

[1] Note that the constituent structures we define here represent the process of parsing rather than the result of parsing.

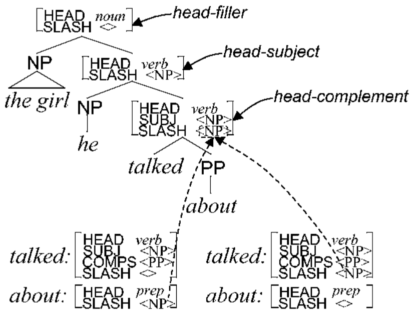

Figure 1: A non-injective grammar rule in HPSG

> *Without loss of generality, we assume $\gamma' = \gamma\gamma''$. By the assumption of the induction, $\exists! c'. c_s \Rightarrow \beta_1 c' \beta_2 \wedge c' \Rightarrow \gamma'$. Since $\rho(\gamma')$ is injective, $\exists! c, c''. c' = \rho(\gamma')(c, c'')$. Hence, $\exists! c, c''. c_s \Rightarrow \beta_1 c c'' \beta_2$. By Definition 3, $c \Rightarrow \gamma$. We thus find that unique c exists for γ.*

The lemma is proved by 1 and 2. According to the lemma, $\forall w \in \mathbf{w}. \exists! c. c_s \Rightarrow \beta_1 c \beta_2 \wedge c \Rightarrow \langle w \rangle$. Hence, the theorem has been proved. □

The theorem shows that we are able to obtain a unique sequence of lexical entries that are consistent with the grammar rules, given the constituent structures and corresponding rule applications.

However, the grammar rules of a lexicalized grammar are not necessarily injective. For example, Figure 1 shows a situation where the SLASH feature of HPSG causes the same parse result for distinct inputs. Phrase "talk about" has an NP in the SLASH feature, but we cannot determine the origination of the NP. A similar argument can be made for LTAG on its adjunction rule. When an auxiliary tree is adjoined into another tree, its spine is melted into the other tree, which produces the same result for distinct inputs.

To enable the acquisition of unique lexical entries even when the grammar rules are not injective, we introduce a further definition that provides a mark which disambiguates the inputs to grammar rules. This allows us to map non-injective rules into injective rules.

Definition 5 (Pseudo-injective) *Grammar rule $r \in R$ is* pseudo-injective *when there exists a function μ such that $r_\mu = \lambda c_1, c_2. \langle r(c_1, c_2), \mu(c_1, c_2) \rangle$ is an injective function. We call μ a marking function, and use R_μ to denote a set of r_μ.*

If all grammar rules are at least pseudo-injective, unique lexical entries are determined. Theorem 1 is now revised into a more general form.

Theorem 2 (Grammar acquisition (revised)) *Given a parse result $c_s \in C$ of a sentence $\mathbf{w}$, a marking function μ, and a derivation history Υ of R_μ, lexical entries for $\mathbf{w}$ are uniquely determined if all grammar rules $r \in R$ are pseudo-injective functions with respect to μ.*

Proof. *We omit the proof since it is much the same as the proof of Theorem 1.* □

Non-injective grammar rules in lexicalized grammars can often be redefined as pseudo-injective functions by the addition of some information, which depends on grammar theories. We now discuss grammar-theory-dependent issues.

CG The injective condition is apparently preserved in a simple CG, which is the class of categorial grammars composed of the two reduction rules: Forward Application (FA) and Backward Application (BA). While these rules take two categories as inputs and output another category, we can regard the rules as taking two derived trees as inputs and outputting a combined tree. With this insight, we can regard reduction rules as grammar rules, derived trees as parse results, derivations as derivation histories, then Theorem 1 can be applied. The annotation cost will not be problematic because we only need to annotate FA or BA to the derived trees by using simple heuristic rules.

This argument is also valid for the extraction of CCG. In order to annotate the other rules defined by CCG (type-raising, composition, and coordination rules), existing work (Hockenmaier and Steedman, 2002) exploits the annotations of traces and coordinations in the Penn Treebank.

LTAG As discussed above, adjoining leads to the situations where the injective condition is violated. We can make the grammar rules pseudo-injective by determining the lengths of the spines, i.e., defining μ as returning the spine length. Existing studies assume the length as one (Xia, 1999), or determine the length by using heuristic rules (Chen and Vijay-Shanker, 2000; Chiang, 2000).

The above discussion was empirically evaluated by acquiring LTAG grammars from the

Grammar (a)		Grammar (b)	
# words	4,514	# words	4,539
# initial trees	1,015	# initial trees	1,068
# aux. trees	1,094	# aux. trees	1,107
coverage	93.4 %	coverage	94.0 %

Table 1: Specifications of the LTAG grammars acquired from the Penn Treebank

Penn Treebank (Marcus et al., 1994). We assumed that the length of all spines was one, i.e., $\forall c_1, c_2. \mu(c_1, c_2) = 1$. 47 heuristic rules were used for the annotation, which shows the annotation was not so costly. A grammar was successfully acquired from Sections 02–21 in 910 seconds on a Xeon 2.0 GHz CPU. As shown in Grammar (a) in Table 1, the grammar had the sufficient coverage (measured against Section 22, 1700 sentences), which shows the robustness of the grammar. We next acquired another grammar (Grammar (b) in Table 1) by including two rules for annotating insertions and coordinations. Since the grammar provides more accurate analyses for insertions and coordinations, the coverage of the grammar has been slightly improved.

HPSG In HPSG, two points lead to violate the injective condition. One is ambiguity in terms of the origin of subcategorization frames as described in Section 3. In Figure 1, SLASH features in HPSG are represented with a set, and the mother's SLASH is the union of the daughters' ones. The union operation is apparently non-injective, but the grammar rules can be made pseudo-injective by defining μ as telling the origin of SLASH features. The other is generalization through the use of a type hierarchy. In HPSG, linguistic generalizations are described by a type hierarchy, where the entities are placed in general/specific relations. Merging of types, *unification*, is inherently not injective. The generalization problem is beyond the scope of our study, and must be left for future research.

4 Conclusion

This study has demonstrated the conditions for acquiring lexicalized grammars from annotated corpora. We proved that a unique lexicalized grammar consistent with the given grammar rules is acquired when derivation histories are given. Our approach enables the development of lexicalized grammars that are robust and broad-coverage, and also lets us compare various grammar theories with real-world texts. This study provides a starting point for the application of linguistic grammar theories to real-world NLP systems.

Future work includes an application of our approach to other grammar theories, such as HPSG. Although annotation of grammar rules (schemata) might be more difficult than LTAG since they have more rules, this will be solved by careful implementation of heuristic annotation rules.

References

John Chen and K. Vijay-Shanker. 2000. Automated extraction of TAGs from the Penn Treebank. In *Proc. 6th IWPT.*

David Chiang. 2000. Statistical parsing with an automatically-extracted tree adjoining grammar. In *Proc. ACL 2000*, pages 456–463.

Julia Hockenmaier and Mark Steedman. 2002. Generative models for statistical parsing with combinatory categorial grammar. In *Proc. 40th ACL*, pages 335–342.

Mitchell Marcus, Grace Kim, Mary Ann Marcinkiewicz, Robert MacIntyre, Ann Bies, Mark Ferguson, Karen Katz, and Britta Schasberger. 1994. The Penn Treebank: Annotating predicate argument structure. In *ARPA Human Language Technology Workshop.*

Philip H. Miller. 2000. *Strong Generative Capacity: The Semantics of Linguistic Formalism.* CSLI Publications.

Ivan A. Sag and Thomas Wasow. 1999. *Syntactic Theory – A Formal Introduction.* CSLI Lcture Notes no. 92. CLSI Publications.

Yves Schabes, Anne Abeillé, and Aravind K. Joshi. 1988. Parsing strategies with 'lexicalized grammars': Application to tree adjoining grammars. In *Proc. 12th COLING*, pages 578–583.

Mark Steedman. 2000. *The Syntactic Process.* The MIT Press.

Fei Xia. 1999. Extracting tree adjoining grammars from bracketed corpora. In *Proc. 5th NLPRS.*