

Using Inverse Lexical Rules to Acquire a Wide-coverage Lexicalized Grammar

Hiroko Nakanishi
University of Tokyo

Yusuke Miyao
University of Tokyo

Jun'ichi Tsujii
University of Tokyo
CREST, JST

Hongo 7-3-1, Bunkyo-ku, Tokyo 113-0033
{n165, yusuke, tsujii}@is.s.u-tokyo.ac.jp

1 Introduction

Automatic grammar extraction from annotated corpora (Xia, 1999; Chen and Vijay-Shanker, 2000; Chiang, 2000; Hockenmaier and Steedman, 2002; Miyao et al., 2004) enabled us to build a wide-coverage lexicalized grammar at low cost. They succeeded in extracting a large number of lexical entries with less effort while conventional methods only allow limited lexical entries to be acquired in real-world texts. Lexicalized grammars require many lexical entries to explain various syntactic alternations, and we can hardly expect that all words will appear in all possible syntactic alternations within a limited training corpus.

We aimed at improving the coverage of an automatically extracted grammar using lexical rules in this work (Jackendoff, 1975; Pollard and Sag, 1994). The idea behind lexical rules is that the syntactic constraints of a group of words are derived with general rules from their *lexemes*, which express characteristics common to the group (e.g. “runs” or “running” is derived from the lexeme “run”).

We automatically acquired lexemes by applying lexical rules *inversely* to the lexical entries of the HPSG grammar extracted automatically from the Penn Treebank (Marcus et al., 1993). We could then generate a wide set of lexical entries from the lexemes, and our grammar achieved a higher coverage against real-world texts.

Although the lexical rules proposed by Pollard and Sag (1994) treated several parts-of-speech such as nouns or adjectives, we only formulated rules for verbs. This is because verbs

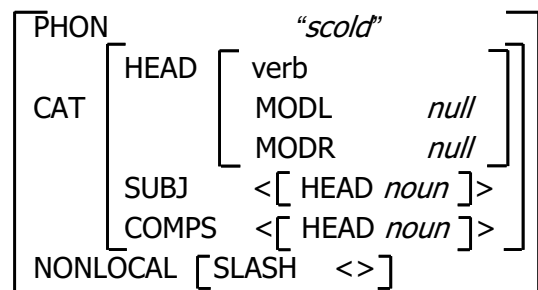


Figure 1: An HPSG lexical entry

have wide variations in syntactic alternations and the effect lexical rules have is significant for verbs.

2 Background

2.1 HPSG

We used an HPSG grammar extracted by the method of Miyao et al. (2004) as input. HPSG (Head-driven Phrase Structure Grammar) provides one of the frameworks for lexicalized grammars. In HPSG, a few grammatical rules called schemata express the general constraints of a grammar and a large number of lexical entries express characteristics specific to words.

Figure 1 outlines an HPSG lexical entry where this is represented by a typed feature structure, which is a set of feature-value pairs. The PHON feature expresses the orthographic information (called a surface) of a word. The CAT expresses a syntactic category. The HEAD feature expresses information inherited from the parent node in the parse tree. MODL and MODR express the constraints of the left and the right modifiers. The SUBJ and the COMPS express lists of features of arguments that

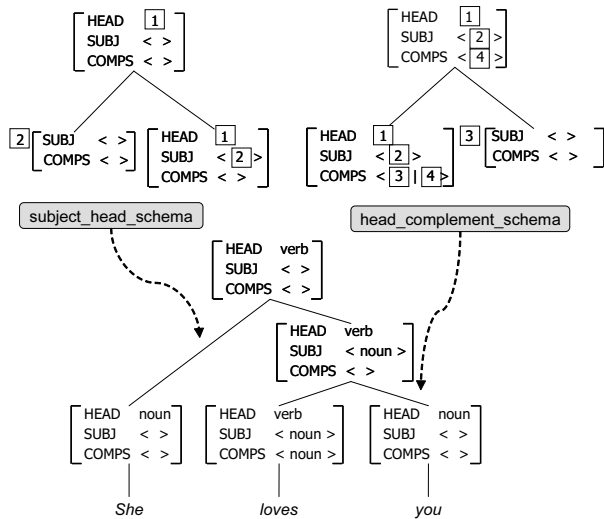


Figure 2: Parsing “She loves you.”

come to the left and right-hand side of a word¹. The NONLOCAL feature expresses unbounded dependency such as in a relative clause or in a WH movement. The features described in Figure 1 are part of those in a lexical entry Miyao et al. (2004) extracted. More detailed constraints can be expressed by increasing features.

Figure 2 shows the process of parsing “She loves you.” with HPSG. First, one of the lexical entries corresponding to each word is chosen and assigned. Next, a suitable schema is chosen in order to combine words. Here, *head_complement_schema* is applied to the lexical entries for “loves” and “you.” Since the feature structure of “you” and the value of the COMPS feature of “loves” are successfully unified, the verb phrase “loves you” is constructed. The value of the HEAD feature of the head “loves” is inherited to the phrase. Finally, through *subject_head_schema*, “she” and “loves you” are combined.

2.2 Automatic extraction of HPSG

Miyao et al. (2004) extracted an HPSG grammar automatically using the Penn Treebank as a training corpus. Their process of grammar extraction was as follows.

Externalization This phase translates annotations in the Penn Treebank into analyses of HPSG by

¹This is not common to all HPSG. In the HPSG for Japanese, both subjects and objects are left arguments.

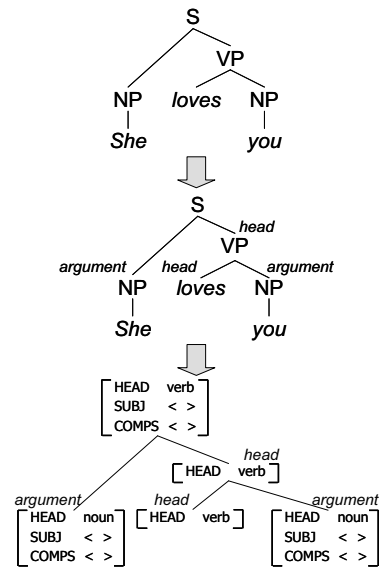


Figure 3: Externalization

adding heuristic annotations. Figure 3 shows the whole process. First, each node of the parse tree is marked *head*, *argument*, or *modifier* using tables representing head-argument pairs² and argument-modifier distinctions³ (Xia, 1999). In Figure 3, “loves” and the VP node above are marked as *head* and NP nodes are marked as *arguments*. Second, the whole tree is rewritten to a binary tree. Then, some heuristic annotations are added in order to analyze special constructions (e.g. coordinations or relative clauses) or fix errors in the Penn Treebank. Finally, an HPSG category is assigned to each non/pre-terminal node of the parse tree according to its pre-terminal or non-terminal symbol. For example, the NP nodes in Figure 3 are assigned a category whose HEAD is *noun*, and SUBJ and COMPS are empty lists.

Extraction In this phase, lexical entries are extracted automatically from the parse tree rewritten in the previous phase. A schema is applied inversely and the child nodes are acquired as the parent node is input. Figure 4 shows the extraction from the sentence “She loves you”. The features not specified in the externalization phase are specified here

²This table lists the arguments each part-of-speech can take (e.g. VB can take NP, PP, and S).

³This table lists the distinctions between arguments and modifiers for each function tags (e.g. -TMP are modifiers, -PRD are arguments).

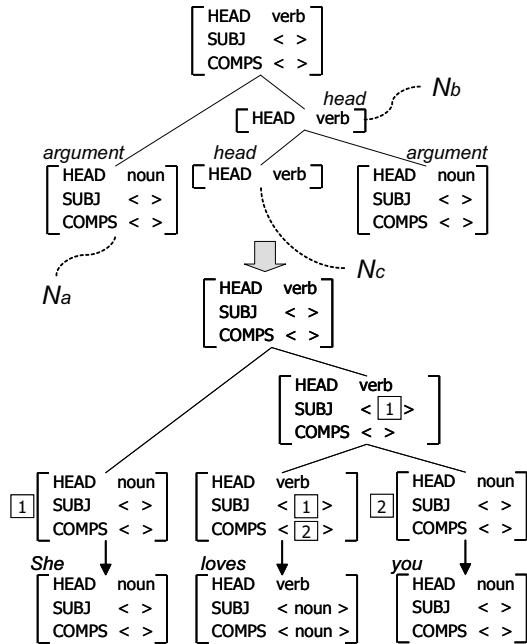


Figure 4: Extraction from “She loves you.”

by schema application. For example, the SUBJ feature in the lexical entry for “loves” are not specified in Figure 3. In application of *subject_head_schema*, the feature structure for the subject is unified with the element of the SUBJ feature of the head (See Figure 2). In this case, the subject is N_a and the head is N_b in Figure 4. The unification determines the SUBJ feature of N_b and the value is inherited to its head child N_c , which is the lexical entry for “loves”. This process is the reverse of the process described in Figure 2.

2.3 Related work

Since lexical entries have more complicated information than counterparts in other lexicalized grammars, the use of lexical rules will greatly influence HPSG. However, we can use our method in other lexicalized grammars such as LTAG or CCG.

There have been several approaches to improve the coverage of a grammar extracted from a corpus. Chen and Vijay-Shanker (2000) classified extracted *elementary trees*, which correspond to lexical entries in HPSG, according to syntactic classes (*tree families*) defined in the manually developed LTAG grammar (the XTAG grammar (XTAG-Group, 1995)). When a word w has an elementary tree which belongs to a certain class, all other elementary trees in

the class are assigned to w . Note that their method can only work when the classification is given. This is just as costly as developing a grammar manually.

Hara et al. (2002) solved this problem by clustering words that had similar elementary trees and acquiring syntactic classes automatically. Our method has an advantage on theirs in that the process to obtain the subcategorization frame of each lexical entry is deterministic owing to manually defined lexical rules that are linguistically motivated. In addition, we can *systematize* the lexicon by linguistic associations given by the lexical rules.

3 Method

Our method consists of the following two steps as outlined in Figure 5.

1. **Reduction** We reduce HPSG lexical entries into their *lexemes* by applying lexical rules inversely to the lexical entries.
2. **Expansion** We expand the acquired *lexemes* into a wide set of lexical entries by applying lexical rules forwardly to lexemes.

When a sentence “The boy is scolded” appears in the training corpus, the conventional method can acquire lexical entry L_1 for “scolded” in the passive voice. However, L_2 for “scolds” in the present tense or L_3 “scolded” in the past tense is not extracted, if the training corpus did not include these usages for “scold”. Hence, the original grammar cannot analyze sentences including such usages for “scold”. In our approach, we assume that all of L_1 , L_2 and L_3 have been derived from a single lexeme L_0 for “scold” through lexical rules. If we could acquire L_0 in reduction, L_2 and L_3 would be acquired by applying lexical rules to L_0 in expansion.

3.1 Definitions

We define a verbal lexeme in HPSG as a kind of a lexical entry that expresses a declarative form of a verb. This definition is different from the definition by Pollard and Sag (1994), where the lexeme “apple” is distinguished from the singular noun “apple” generated from it. Our definition basically follows the definition of “base trees” (lexemes in LTAG) by Prolo (2002). Formally, given set Ψ of lexical entries, set Φ of lexemes is a subset of Ψ , i.e., $\Phi \subset \Psi$.

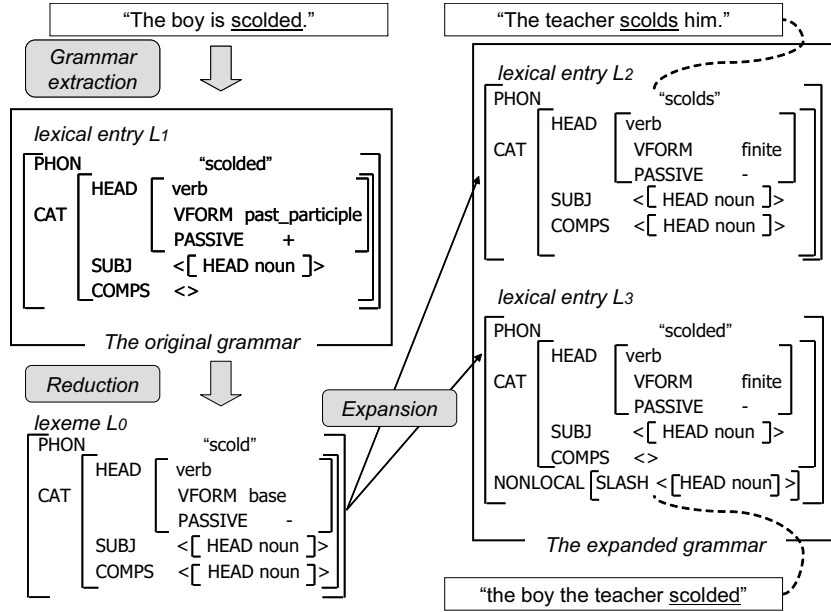


Figure 5: Reduction and expansion

We define a lexical rule as function $\rho : \Psi \rightarrow \Psi$ and a set of lexical rules as R . The rule corresponding to $X \rightarrow Y$ is function ρ , which satisfies $\rho(X) = Y$. The inverse lexical rule of ρ is denoted as $\rho^i : \Psi \rightarrow \Psi$, which satisfies $\rho^i(\rho(X)) = X$. We denote a set of inverse lexical rules as R^i .

We manually implemented 24 lexical rules and their inverse lexical rules (Table 1). Some of them are *derivational rules*, which have no inflection. To capture the characteristics of the original grammar, some of the rules proposed by (Pollard and Sag, 1994) were omitted and some other rules were added. For example, one of the omitted rules was *negation_rule*, which generates a verb in a negative sentence. This is because verbal lexical entries in a positive sentence and those in a negative sentence are not distinguished in the original grammar. However, we added several rules to create participial constructions. Participial constructions have various patterns determined by whether they have a subject or not, whether the main clause is on the right or left, and whether they are headed by a present or past participle. We implemented eight rules to capture all of those.

3.2 Reduction

This phase acquires lexemes by applying inverse lexical rules to lexical entries in the grammar.

Rule	Example
inflectional rule	
singular3rd	He loves Mary.
no_singular3rd	I love Mary.
past	He loved Mary.
present_participle	He is eating breakfast.
gerund	He finished reading the book.
passive	Mary is loved by him.
passive_r(l)_adjunct	fried chicken(right)
prp_r(l)_adjunct	a singing bird(right)
perfect	He has finished his homework.
participial_construction	Seeing a bear, they ran away.
past_modal	He might have lost his way.
derivational rule	
drop_by	Mary is loved .
wh_move	the work he has to do
imperative	Give it to me.
yn_question	Can you open the door?

Table 1: Lexical rules

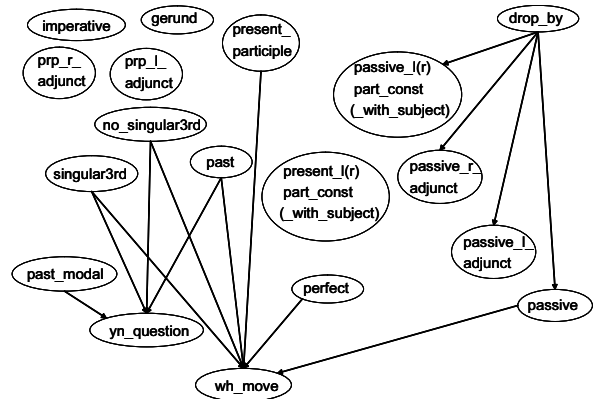


Figure 6: Constraints between lexical rules

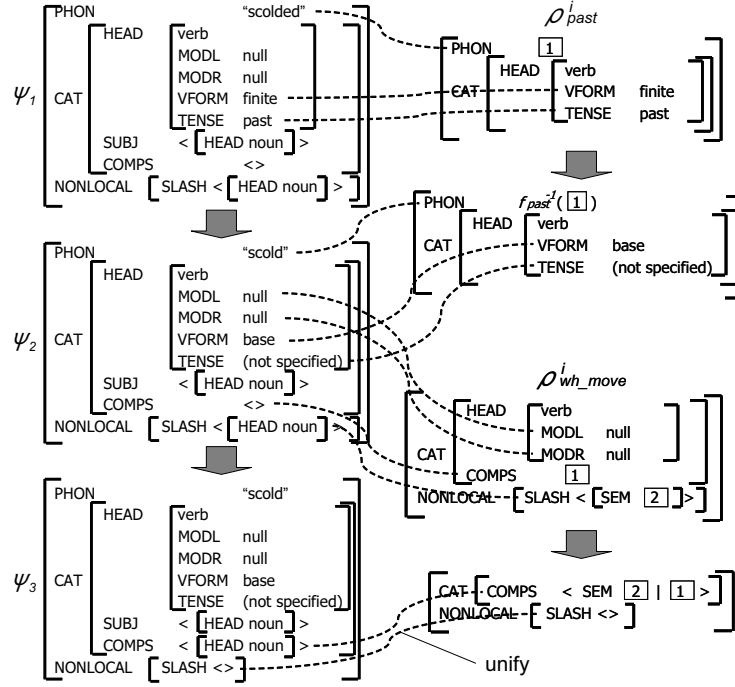


Figure 7: Acquiring lexeme of “scolded”

We defined a partial order $\langle R, \preceq \rangle$ on the basis of the linear order by Prolo (2002) to avoid invalid combinations of rules. The graph in Figure 6 shows constraints between lexical rules, where $\rho_1 \rightarrow \rho_2$ means $\rho_1 \preceq \rho_2$. After we apply rule ρ_1^i to lexical entry ψ_1 , we can apply another rule ρ_2^i to $\rho_1^i(\psi_1)$ only when $\rho_2 \preceq \rho_1$. If these constraints had not been defined, inverse lexical rules could be combined freely to produce invalid lexical entries. For example, the combination of $\rho_{imperative}^i$ and $\rho_{passive}^i$ is impossible in real texts. This partial order is necessary to avoid these kinds of meaningless combinations of rules.

Figure 7 shows the process for acquiring a lexeme. Input is the lexical entry ψ_1 for “scolded”, which is extracted from the clause “the boy the teacher scolded”. First, ρ_{past}^i is applied to ψ_1 . The values for the VFORM and TENSE features of ψ_1 are successfully unified with those for the input of the rule. In $\psi_2 = \rho_{past}^i(\psi_1)$, the values for the VFORM and TENSE are rewritten according to the constraints described in the rule. Function f_{past}^i reduces an inflection for past tense. The surface “scolded” is converted to “scold” with this function. The other values, which are not mentioned in the rule (e.g. MODL and MODR), remain as they were.

Next, $\rho_{wh_move}^i$ is applied to ψ_2 . The values for the SLASH feature of ψ_2 and the input of the rule are unified and the value becomes the first element of the COMPS feature of output. Finally, $\psi_3 \in \Phi$ is acquired as the output of reduction.

If input ψ is successfully reduced into lexeme ϕ , we register ϕ into the database. Otherwise, we register ψ . As a result, the database includes both lexemes and lexical entries that cannot be reduced by inverse lexical rules.

3.3 Expansion

This phase is dynamically executed when we parse texts with the grammar. We look up a lexeme for each word and assign the lexical entries expanded from the lexeme by applying lexical rules to the word.

4 Experimental results

We applied our algorithm to all verbal lexical entries extracted from Section 02-21 of the Penn Treebank (9415 words, 1487 lexical entry templates⁴). As a result of reduction, 5029 words and 1067 lex-

⁴A lexical entry template is a lexical entry whose PHON feature is removed. A lexical entry is represented with a pair of a word and a lexical entry template.

		seen (sw, sc)	unseen				# of lexical entries
			(sw, sc)	(sw, uc)	(uw, sc)	(uw, uc)	
G_0	all	95.01	2.19	0.08	2.71	0.004	29.32
	verbs	91.88	5.32	0.47	2.31	0.02	-
G_1	all	95.45	1.89	0.08	2.58	0.002	41.93
	verbs	94.62	3.41	0.45	1.51	0.01	-
G_2	all	95.71	1.77	0.07	2.44	0.002	55.62
	verbs	96.20	2.68	0.42	0.68	0.01	-

Table 2: Lexical coverage against Section 23

ical entry templates were obtained. Their 265 templates corresponded to lexemes. Application of inverse lexical rules failed for 802 lexical entry templates, and they remained as exceptional forms. This indicates that our implementation of lexical rules required further enhancement to capture the syntactic alternations that occur in real-world texts.

To evaluate the impact our approach had on the coverage of real-world texts, we measured the lexical coverage achieved by the original and our grammar. Table 2 lists the lexical coverage against Section 23. The “seen” column is the ratio the exact pair (word, lexical entry template) is found in the database. The “unseen” column is the ratio the pair is not found. The “sw/uw” and “sc/uc” in the “unseen” column mean seen/unseen words and seen/unseen templates. The “all/verb” rows express the lexical coverage of all lexical entries/verbs. G_0 is the original grammar. G_1 is a grammar developed by only applying the expansion phase. This was expanded using the 180 templates for lexemes that originally existed in G_0 . G_2 is a grammar developed by applying both reduction and expansion. The results show that G_2 obtained 96.20% coverage against verbs, which is the highest of the three. This means that the inverse lexical rules in the reduction phase greatly contributed to improving the coverage. The “# of lexical entries” column expresses the average number of lexical entries assigned to one word. G_2 assigned about 2.7 times as many lexical entries as G_0 . However, it is not clear whether all lexical entries in G_2 contributed to improving the coverage. In another experiment, we investigated the possibility that our algorithm generated unnecessary lexical entries.

“ G_0 ” and “ G_2 ” in Figure 8 indicate improvements in the lexical coverage of verbs when the size of

the training corpus increased. This indicates that G_2 could acquire high coverage with a small corpus thanks to reduction and expansion. While the increased coverage of G_0 did not saturate, that of G_2 became quite slow after 10000 sentences. When a seen word in a certain syntactic alternation was found in the corpus, G_0 regarded it as a new lexical entry if the word had not yet appeared in the same syntactic alternation. Even then, G_2 regarded it as a known lexical entry if the lexeme of this word was acquired and the lexical rule expressing the syntactic alternation was supported. This is why the increase in coverage slowed down in G_2 . “ G_0 ” and “ G_2 ” in Figure 9 indicate the number of lexical entries under the same conditions as in Figure 8. In contrast to the results in coverage, the increase in G_2 was much faster than that in G_0 . This is because the appearance of a new lexeme could lead to dozens of lexical entries to being assigned one word in G_2 . These results imply that lexemes acquired later contributed little to coverage. Unnecessary lexical entries can be generated from such lexemes. Additionally, assigning too many lexical entries to one word may slow down the parsing process.

To avoid unnecessary expansion, we set a threshold for the frequencies of lexemes in the training corpus. We expanded a lexeme only when its frequency was above a certain threshold. In Figure 8 and Figure 9, “threshold= x ” indicates the increase in coverage of verbs and the number of lexical entries at several thresholds. Since the results for thresholds of 50 and 100 were almost the same, 50 was considered to be the appropriate in our setting.

Note that the coverage of verbs is saturated at about 96%. The remaining 4% does not seem to be covered by our method because we cannot generate enough variations in lexical entries for words that

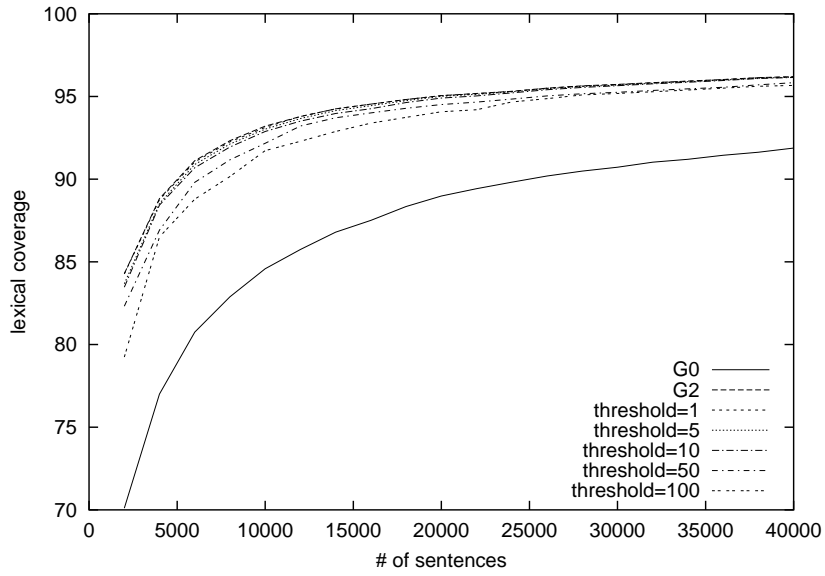


Figure 8: Lexical coverage of verbs

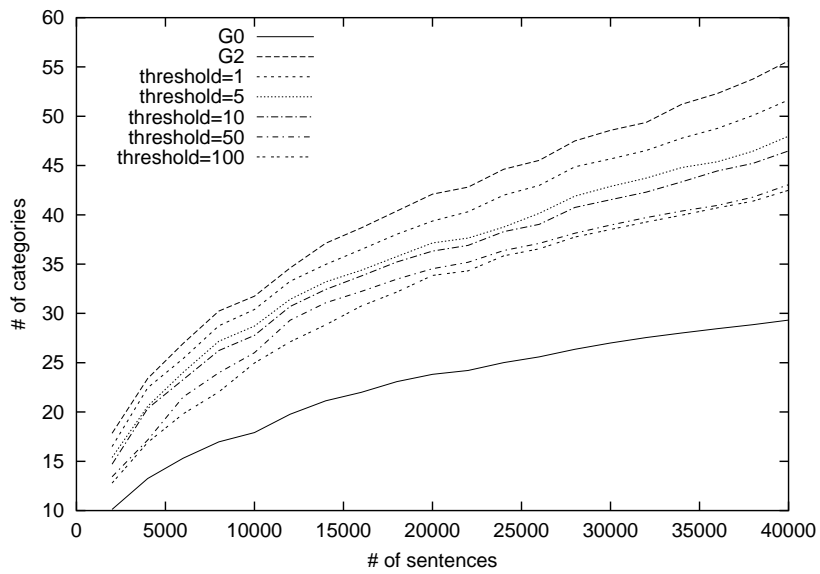


Figure 9: Average number of lexical entries

never appeared or appeared just once in the training corpus. This problem could be solved by using a method of treating unknown words (Hockenmaier and Steedman, 2002) appropriately.

5 Conclusion

We proposed a method of improving the coverage of a lexicalized grammar using inverse lexical rules. Our experiments revealed that the grammar obtained with our algorithm achieved higher coverage with a smaller corpus than the original grammar did. Although this improvement resulted from the increased number of lexemes, too much expansion generated lexical entries that made little contribution to coverage. Setting an appropriate threshold was useful to avoid generating such lexical entries. In addition, we have to refine lexical rules to capture syntactic alternations precisely.

References

- John Chen and K. Vijay-Shanker. 2000. Automated extraction of TAGs from the Penn Treebank. In *Proceedings of 6th IWPT*.
- David Chiang. 2000. Statistical parsing with an automatically-extracted tree adjoining grammar. In *Proceedings of 38th ACL*, pages 456–463.
- Tadayoshi Hara, Yusuke Miyao, and Jun'ichi Tsujii. 2002. Clustering for obtaining syntactic classes of words from automatically extracted LTAG grammars. In *Proceedings of the sixth International Workshop on Tree Adjoining Grammars and Related Frameworks*, pages 227–233.
- Julia Hockenmaier and Mark Steedman. 2002. Acquiring compact lexicalized grammars from a cleaner treebank. In *Proceedings of Third International Conference on Language Resources and Evaluation*.
- Ray S. Jackendoff. 1975. Morphological and semantic regularities in the lexicon. *Language*, 51:639–671.
- Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics*.
- Yusuke Miyao, Takashi Ninomiya, and Jun'ichi Tsujii. 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*.
- C. Pollard and Ivan A. Sag. 1994. Head-driven Phrase Structure Grammar.
- Carlos A. Prolo. 2002. Generating the XTAG English grammar using metarules. In *Proceedings of the 19th International Conference on Computational Linguistics (COLING'2002)*, pages 814–820.
- Fei Xia. 1999. Extracting tree adjoining grammars from bracketed corpora. In *Proceedings of 5th NLPRS*.
- The XTAG-Group. 1995. A Lexicalized Tree Adjoining Grammar for English. Technical report, IRCS 95-03.